GAMS/MPSX INTERFACE

by

LAWRENCE E. MCPHAIL

Prepared for:
Dr. Richard Barr
OREM 4390
Senior Design

DEPARTMENT OF OPERATIONS RESEARCH AND ENGINEERING MANAGEMENT
SCHOOL OF ENGINEERING AND APPLIED SCIENCE
DALLAS, TEXAS 75275
GAMS/MPSX INTERFACE

by

LAWRENCE E. MCPHAIL

Prepared for:
Dr. Richard Barr
OREM 4390
Senior Design

ABSTRACT

This report briefly describes the General Algebraic Modeling System (GAMS), and some of its commands. Also, integer programming is discussed. Finally, a description of the implementation process for the GAMS/MPSX interface is presented.

10 May 1986
# Table of Contents

## Part I: Introduction to GAMS

- General form of the GAMS statement
  - GAMS Input
  - SETS Keyword
  - DATA
  - SOLUTION
  - GAMS Output
  - Echo Print
  - Symbol Listing
  - Equation Listing
  - Column Listing
  - Model Statistics
  - Solution Report

## Part II: Integer Programming

## Part III: The Interface to MPSX/370

## Summary
INTRODUCTION

The General Algebraic Modeling System (GAMS) allows the user to concisely state his models in a language that is easily understood by both the computer and the modeler. The subject of this paper is first, the GAMS language, and second, the implementation of an interface between GAMS and the IBM Mixed Integer Programming/370 (MIP/370), which is part of the IBM Mathematical Programming System Extended/370 (MPSX/370), to allow GAMS to access the integer programming power of MIP/370. In this paper, we will show what has been done to implement this interface, and what has yet to be accomplished.

The explanation of GAMS in this paper, and the discussion of Integer Programming and the computer interface to MPSX/370, assumes that the reader is familiar with the basics of modeling systems and linear programming. The discussion of the computer interface will assume some prior knowledge of computer systems.
PART I: INTRODUCTION TO GAMS

The General Algebraic Modeling System, or GAMS, was first in use at the International Bank for Research and Development (the World Bank) in 1980. Alexander Meeraus, responsible for much of the computer programming at the World Bank, realized that a modeling system could eliminate many of the problems that the economists and computer programmers at the World Bank were having with developing optimization models and large simulations of different sectors of world industry. The economists would not recognize their model once it had been put into a form that the computers could understand, and the programmers spent long hours finding errors in the FORTRAN programs that they wrote to solve each economists model.

The prototype of GAMS was in use at the World Bank, and also at the University of Texas at Austin, where David Kendrick taught economic modeling to graduate students. GAMS has still not been officially released, but its success at these two, and other institutions (such as SMU), has been documented.

This introduction to GAMS will be very brief, and is based on GAMS: An Introduction, by Kendrick and Meeraus. The text that we have used is a February 1985 draft of the manual, as presented to Dr. Richard Barr, SMU department of Operations Research and Engineering Management. Unless otherwise noted, all reference in this section is drawn from this text.

GENERAL FORM OF THE GAMS STATEMENT

The general form of the GAMS statement is this:

(keyword) (declaration) (definition)

where the keyword is one of the recognized keywords from the set of GAMS keywords such as SET, PARAMETER, SCALAR, VARIABLES, etc. The declaration pertains to the command that has been issued, and usually contains a name, and a description or range for the operator that is being used. The definition portion of the keyword includes such items as a list of elements in a set, the value of a variable or list of variables, or the elements of a matrix of constants (used with the TABLE keyword).

GAMS INPUT

The input to GAMS takes several forms:
SETS

Sets can be used to make a model statement easy to read. They are an efficient way to state many economic models. The **SET** keyword is used in this manner:

```gams
SET PRICES PRICES OF COMMODITIES
/GOLD-PRICE GOLD DOLLARS PER OUNCE
   SIL-PRICE SILVER DOLLARS PER OUNCE/
```

The keywords all start anywhere but in the first column, which is reserved for comments and print commands. The descriptor (in this case, **PRICES OF COMMODITIES**) is optional. The identifier (**PRICES**) can be up to ten characters long, and can not contain any embedded blanks. The elements of the set (also called labels) can be up to ten characters long with no embedded spaces, and may also have descriptors associated with them.

DATA

Data element statements take several form, based on the data being provided. The **SCALAR** keyword identifies constants that are to be used in the model. The keyword **PARAMETER** signifies a list of constants. For two-way, or matrix, constants, the keyword is **TABLE**. All names of definitions for constants allow descriptors as well.

EQUATIONS AND MODELING CONSTRAINTS

GAMS provides for equality and inequality equations, using the symbols **=E=**, **=G=**, and **=L=**. The constraints can be labeled with text descriptors, for use in the report, such as **AVAILABILITY CONSTRAINT**, or **SUPPLY CONSTRAINT**. GAMS allows the user to specify equations to be used during solution, although the common entry is **/ALL/**.

SOLUTION

The GAMS language provides for several solution parameters to be supplied by the user. GAMS also allows several **SOLVE** statements, allowing the model to be evaluated over several conditions, or using different equations. This sample of the **SOLVE** statement is taken from the **Newhaven Busing Proposal Case**:

```gams
SOLVE NEWHAVEN USING LP MINIMIZING TOTDIST
```

In this statement, the user tells GAMS to solve the model called **NEWHAVEN** using Linear Programming methods with the objective of minimizing the variable **TOTDIST**, which the user has described as "the total student-miles
traveled."

There is a statement available to the user that we will discuss primarily because our project will be concerned with using this statement. The **OPTION** statement provides the user with the capability to tell GAMS to use a different optimizer or system package to arrive at the solution. For instance, the statement could read:

```
OPTION LP = MINOS5
```

which instructs GAMS to use the MINOS system. We will implement an interface package that will allow the user to specify:

```
OPTION MP (or RMIP) = MPSX
```

This will allow GAMS to access the IBM Mixed Integer Programming/370 package on the MPSX/370 system to solve mixed integer programming problems. **RMIP** stands for **Relaxed Mixed Integer Programming**, which simply means that the constraints requiring a variable to be of the type **INTEGER** are ignored. We will discuss this interface in part three of this report.

**GAMS OUTPUT**

The GAMS output comes in several parts. They are:

**ECHO PRINT**

The **Echo Print** is simply a listing of your GAMS data file, except that it has line numbers associated with the statements, and has the title that you specified for your model printed in the upper left hand corner of the page (as do all subsequent pages). The echo print does not introduce any new information.

**SYMBOL LISTING**

The next output item, the **Symbol Listing**, is a reference summary of all of the data names, variables, and equations used in the model. Each symbol is listed in a table, along with its type and any references to it, and is then grouped with similar type symbols and printed with its text description. This listing is a useful reference for the user as to all symbols used in the model.

**EQUATION LISTING**

The **Equation Listing** provides a reference to the equations used, their type, and the text description. This listing also replaces data names with the actual
values, so that the user can refer to the equation in a form that he is used to seeing it in. This also puts all equations into standard form.

COLUMN LISTING

This portion of the output gives part of the columnar listing for the variables involved in the model.

MODEL STATISTICS

Model Statistics provides the user with information about the model itself, such as numbers of major and minor rows and columns, the time it took to generate the model, and the execution time, among other data.

SOLUTION REPORT

The Solution Report is the part of the output that tells the user what was determined about the model and the variables. The report starts with a summary of the solution, then gives the user information about the constraints, definition equations, and finally, if the model was solved, the optimal values for the variables. If the model was insoluble, had non-optimal solutions, was unbounded, or infeasible, the solution report would give the user information to this effect as well.

This was a brief introduction to the GAMS language itself. For a detailed reading, contact Dr. Barr for a copy of the report by Meeraus and Kendrick.
In most applications, the preferred solution to a linear programming model would have variables with integer values. If you presented Lee Iacocca with a recommendation to build 576,972.42 Chryslers, nobody would much notice if he rounded the number up or down. However, if you presented a plan to a city manager to build 1.42 new schools, a lot of construction companies, taxpayers, and parents would be very interested in the rounding of the number. In general, if the solution is relatively small, one would prefer integer values for the variables. In the case of large solutions, the effect of rounding a continuous variable is less noticeable. Of course, one can come up with exceptions to this, as any, rule.

There are several classes of integer programming. There is pure integer or all-integer problem is one in which all variables are integer variables. In mixed integer problems, some, but not all, of the variables are integer variables. In 0/1 problems, the variable can only take on a value of zero or one. This is in contrast to general integer problems, which simply restrict the variable to an integer value in any feasible solution.

In continuous linear programs (that is, linear programs without integer variables), the solution time generally increases fairly proportionally to the number of variables, and approximately with the square of the number of constraints. Integer programs, however, may actually solve quicker with additional constraints, or these additional constraints may dramatically increase the amount of time necessary to arrive at a solution.

The following example, from the book *Linear, Integer, and Quadratic Programming with LINDO*, by Linus Schrage is an example of a 0/1 Integer Program (the LINDO solution is in the appendix at the end of this report):

\[
\begin{align*}
\text{MAX} & \quad 75X_1 + 6X_2 + 3X_3 + 33X_4 \\
\text{SUBJECT TO} & \quad 2) \quad 774X_1 + 76X_2 + 22X_3 + 42X_4 \leq 875 \\
& \quad 3) \quad 67X_1 + 27X_2 + 794X_3 + 53X_4 \leq 875 \\
X_1, X_2, X_3, X_4 \text{ RESTRICTED TO } 0, 1.
\end{align*}
\]

This example was used by Schrage to illustrate the Branch and Bound technique of solving integer programs.

Other applications of integer programs include capital investments, fixed charge, cost curves (with economies
Several algorithms have been developed for solving integer problems, although the Branch and Bound method has proven most reliable, and virtually all commercial code uses this method to solve integer programs. The figure below illustrates branch and bound methodology.

Other algorithms include Gomory's Cut Algorithms, including The Gomory fractional algorithm, The Gomory all-integer algorithm, and The Gomory mixed-integer algorithm; The Intersection Cut, Young's Primal Algorithm, and Glover's Primal Algorithm, among others.

(This chart is from ZIONTS, page 416)
Originally, this project was to design and implement an interface from GAMS to MPSX. This turned out to be much harder than first anticipated. Documentation of the file structures for output files from MPSX was negligible. Then, Dr. Barr discovered that the interface had already been done at the Computing Center at the University of Texas/Austin, so the project evolved to its present state of implementation of the Austin package on the VM system here at Southern Methodist University. Dr. Barr received the files that were necessary from Chung-Hsing Huang (Steve), a graduate student at UT/Austin. These consisted of seventeen files: seven help files, a PASCAL program, a FORTRAN program, a data file, and seven EXEC, or program control, files. Initially, after receiving approval from the SMU Computing Center to go ahead with the project, the PASCAL program was compiled and indicated numerous errors. We inspected the source code, and determined that the majority of the errors were caused by a difference in the default margins of our compiler and the UT/Austin compiler. After correcting this error, the PASCAL program compiled without error. The next step was to work on the EXEC, or control, files. EXEC files control the flow of the program between different packages on the system, and allow the programmer to access software in other areas of the computer's virtual and disk memory. This allows the program to run in a manner termed invisible to the user of the software package; that is, the user is unaware of the instructions that the computer has performed to access various software packages, and makes it unnecessary for the user to perform these commands. Upon inspection, it was determined that modifications would have to be made to allow for the fact that the UT/Austin system had the various packages stored on different disks than the VM system here at SMU. This was expected, and constitutes the personalization of the software to conform to our system specifications. Upon the completion of the modifications to the EXEC files, the system package will be tested, and on successful completion of the tests, added to the VM system programs, available to all users. This paper can not go into the details of inspection and debugging or modification of the source code and EXEC files, due to lack of space and minimal interest in the procedure.

Once the system is available to all users, it will be accessed as described in Part I, through an invocation of the OPTIONS statement:

```
OPTIONS MP (or RMIP) = MPSX
```
This specification of the MPSX package will invoke the MPSX EXEC file, which tells the computer how to go about using the MPSX package to solve the problem. The GAMS information is translated into a form that MPSX understands, and MPSX operates upon the data, creating an output file, which is translated back into a format that can be used by GAMS to produce the GAMS report as earlier described. The user will see information on the screen telling him what process the system is in, but he will not be required to use any system commands to activate the process. Once the GAMS documentation is available to people using the system, documentation for the MPSX interface will be included to instruct users in the correct usage of the package. This package will be of special interest to the faculty and students in the OREM department, and could be used in classes on linear and integer programming and for projects and assignments in the coming years.
SUMMARY

The concise, friendly modeling language, GAMS, can be used to develop and solve models in a way that both the computer and the modeler can easily understand. The consistent statement structure, convenient output summaries and reports, and the "plain English" type source code make for a system that is easy to understand and use, and yet provides extremely useful information.

Integer Programming involves pure (or all) integer programming, mixed-integer programming, 0/1 problems, and general integer problems. The most common technique for solving integer programs in commercial code is the Branch and Bound technique.

The GAMS/MPSX interface involves the inspection and preparation of PASCAL and FORTRAN programs, and EXEC, or program control, files. The package was received from the University of Texas at Austin, and implemented on the VM system here at Southern Methodist University. Documentation will be provided when a GAMS manual is put together for the users on the SMU VM system. This package is ideal for use in OREM classes on linear and integer programming, and for projects and assignments in the future.
SOURCES

Linear and Integer Programming

Linear, Integer, and Quadratic Programming with LINDO

GAMS: An Introduction

FOOTNOTES

1. SCHRAGE, page 186.

2. IBID., page 191.

3. ZIONTS, pages 343-370

Throughout the report, various ideas have been drawn from all three works. These distillations have not been attributed, due to the combinatory and random reference to the ideas, rather than the words of the writings.
APPENDIX

A. EXAMPLE GAMS PROBLEM AND SOLUTION
   1. Data File
   2. GAMS Output

B. LINDO EXAMPLE (INTEGER PROGRAMMING)

C. LISTINGS OF INTERFACE FILES
   1. EXEC Files
   2. Data File
   3. Programs
THE FILES IN THIS APPENDIX (WITH THE EXCEPTION OF TEMPSX PASCAL) ARE IN THE
ORIGINAL FORMAT IN WHICH THEY WERE RECEIVED FROM UNIVERSITY OF TEXAS AUSTIN.

IF ONE WOULD ATTEMPT TO RUN THESE FILES, OR USE THEM IN ANY WAY, THE BENEFIT
OF THE PERSONALIZATION PROCESS FOR SOFTWARE WOULD BE IMMEDIATELY APPARENT.

IF THERE ARE ANY QUESTIONS, PLEASE DIRECT INQUIRIES TO:

LAWRENCE E. MCPHAIL
C/O PROMAX ASSOCIATES
5015 TRACY STREET
SUITE 203
DALLAS, TEXAS 75205
(214) 522-2449
STITLE A TRANSPORTATION PROBLEM (CANNERY, SEQ=1)

THIS PROBLEM FINDS A LEAST COST SHIPPING SCHEDULE THAT
MEETS REQUIREMENTS AT WAREHOUSES AND SUPPLIES AT CANNERIES.

REFERENCE: G. B. DANTZIG, LINEAR PROGRAMMING AND EXTENSIONS,
PRINCETON UNIVERSITY PRESS, PRINCETON, NEW JERSEY, 1963.

SETS I CANNERIES / SEATTLE, SAN-DIEGO /
J WAREHOUSES / NEW-YORK, CHICAGO, KANSAS-CTY /
PARAMETER A(I) AVAILABLE INVENTORY (CASES OF TINS PER YEAR) /
  SEATTLE 350
  SAN-DIEGO 600 /
PARAMETER R(J) REQUIRED INVENTORY (CASES OF TINS PER YEAR) ;
  R(J) = 300;
TABLE C(I,J) UNIT TRANSPORT COST (DOLLARS PER CASE)
     NEW-YORK  CHICAGO  KANSAS-CTY
SEATTLE     25       17       18
SAN-DIEGO   25       18       14
VARIABLES X(I,J) SHIPMENTS (CASES OF TINS PER YEAR)
TRCOST       TRANSPORT COST (DOLLARS PER YEAR)
POSITIVE VARIABLE X;
EQUATIONS SUPPLY(I) AVAILABILITY CONSTRAINT
DEMAND(J) REQUIREMENTS CONSTRAINT
COST COST DEFINITION ;
SUPPLY(I) .. SUM(J, X(I,J)) =L= A(I) ;
DEMAND(J) .. SUM(I, X(I,J)) =G= R(J) ;
COST.. TRCOST =E= SUM((I,J), C(I,J)*X(I,J)) ;
MODEL CANNERY /ALL/ ;
SOLVE CANNERY USING LP MINIMIZING TRCOST ;
A TRANSPORTATION PROBLEM (CANNERY,SEQ=1)

• THIS PROBLEM FINDS A LEAST COST SHIPPING SCHEDULE THAT
• MEETS REQUIREMENTS AT WAREHOUSES AND SUPPLIES AT CANNERIES.

• REFERENCE: G B DANTZIG, LINEAR PROGRAMMING AND EXTENSIONS,
  PRINCETON UNIVERSITY PRESS, PRINCETON, NEW JERSEY, 1963.

SETS I CANNERIES / SEATTLE, SAN-DIEGO /
  J WAREHOUSES / NEW-YORK, CHICAGO, KANSAS-CTY /

PARAMETER A(I) AVAILABLE INVENTORY (CASES OF TINS PER YEAR) /
  SEATTLE 350
  SAN-DIEGO 600 /

PARAMETER R(J) REQUIRED INVENTORY (CASE OF TINS PER YEAR) ;
  R(J) = 300;

TABLE C(I,J) UNIT TRANSPORT COST (DOLLARS PER CASE)
  NEW-YORK CHICAGO KANSAS-CTY
  SEATTLE  25  17  18
  SAN-DIEGO 25  18  14

VARIABLES X(I,J) SHIPMENTS (CASES OF TINS PER YEAR)

TRCOST TRANSPORT COST (DOLLARS PER YEAR)

POSITIVE VARIABLE X;

EQUATIONS SUPPLY(I) AVAILABILITY CONSTRAINT
  DEMAND(J) REQUIREMENTS CONSTRAINT
  COST COST DEFINITION ;

SUPPLY(I) .. SUM(J, X(I,J)) =L= A(I) ;

DEMAND(J) .. SUM(I, X(I,J)) =G= R(J) ;

COST.. TRCOST =E= SUM((I,J), C(I,J)*X(I,J)) ;

MODEL CANNERY /ALL/ ;

SOLVE CANNERY USING LP MINIMIZING TRCOST ;
### A TRANSPORTATION PROBLEM (CANNERY,SEQ=1)

SYMBOL LISTING

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>TYPE</th>
<th>REFERENCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>PARAM DECLARED</td>
<td>12 DEFINED</td>
</tr>
<tr>
<td>C</td>
<td>PARAM DECLARED</td>
<td>21 DEFINED</td>
</tr>
<tr>
<td>CANNERY</td>
<td>MODEL DECLARED</td>
<td>41 DEFINED</td>
</tr>
<tr>
<td>COST</td>
<td>EQU DECLARED</td>
<td>33 DEFINED</td>
</tr>
<tr>
<td>DEMAND</td>
<td>EQU DECLARED</td>
<td>32 DEFINED</td>
</tr>
<tr>
<td>I</td>
<td>SET DECLARED</td>
<td>9 DEFINED</td>
</tr>
<tr>
<td>J</td>
<td>SET DECLARED</td>
<td>10 DEFINED</td>
</tr>
<tr>
<td>R</td>
<td>PARAM DECLARED</td>
<td>17 ASSIGNED</td>
</tr>
<tr>
<td>SUPPLY</td>
<td>EQU DECLARED</td>
<td>31 DEFINED</td>
</tr>
<tr>
<td>TRCOST</td>
<td>VAR DECLARED</td>
<td>28 IMPL-ASN</td>
</tr>
<tr>
<td>X</td>
<td>VAR DECLARED</td>
<td>27 IMPL-ASN</td>
</tr>
</tbody>
</table>

### SETS

<table>
<thead>
<tr>
<th>SET</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>CANNERIES</td>
</tr>
<tr>
<td>J</td>
<td>WAREHOUSES</td>
</tr>
</tbody>
</table>

### PARAMETERS

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>AVAILABLE INVENTORY (CASES OF TINS PER YEAR)</td>
</tr>
<tr>
<td>C</td>
<td>UNIT TRANSPORT COST (DOLLARS PER CASE)</td>
</tr>
<tr>
<td>R</td>
<td>REQUIRED INVENTORY (CASES OF TINS PER YEAR)</td>
</tr>
</tbody>
</table>

### VARIABLES

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRCOST</td>
<td>TRANSPORT COST (DOLLARS PER YEAR)</td>
</tr>
<tr>
<td>X</td>
<td>SHIPMENTS (CASES OF TINS PER YEAR)</td>
</tr>
</tbody>
</table>

### EQUATIONS

<table>
<thead>
<tr>
<th>EQUATION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>COST</td>
<td>COST DEFINITION</td>
</tr>
<tr>
<td>DEMAND</td>
<td>REQUIREMENTS CONSTRAINT</td>
</tr>
<tr>
<td>SUPPLY</td>
<td>AVAILABILITY CONSTRAINT</td>
</tr>
</tbody>
</table>

### MODELS

<table>
<thead>
<tr>
<th>MODEL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CANNERY</td>
<td></td>
</tr>
</tbody>
</table>

### Compilation Time

COMPILATION TIME = 0.313 SECONDS
A TRANSPORTATION PROBLEM (CANNERY,SEQ=1)
EQUATION LISTING  SOLVE CANNERY USING LP FROM LINE 43

----- SUPPLY =L= AVAILABILITY CONSTRAINT

SUPPLY(SEATTLE).. X(SEATTLE,NEW-YORK) + X(SEATTLE,CHICAGO) + X(SEATTLE,KANSAS-CTY) =L= 350 ;

SUPPLY(SAN-DIEGO).. X(SAN-DIEGO,NEW-YORK) + X(SAN-DIEGO,CHICAGO) + X(SAN-DIEGO,KANSAS-CTY) =L= 600 ;

----- DEMAND =G= REQUIREMENTS CONSTRAINT

DEMAND(NEW-YORK).. X(SEATTLE,NEW-YORK) + X(SAN-DIEGO,NEW-YORK) =G= 300 ;

DEMAND(CHICAGO).. X(SEATTLE,CHICAGO) + X(SAN-DIEGO,CHICAGO) =G= 300 ;

DEMAND(KANSAS-CTY).. X(SEATTLE,KANSAS-CTY) + X(SAN-DIEGO,KANSAS-CTY) =G= 300 ;

----- COST =E= COST DEFINITION

COST.. -25*X(SEATTLE,NEW-YORK) - 17*X(SEATTLE,CHICAGO) - 18*X(SEATTLE,KANSAS-CTY) - 25*X(SAN-DIEGO,NEW-YORK)
- 18*X(SAN-DIEGO,CHICAGO) - 14*X(SAN-DIEGO,KANSAS-CTY) + TCOST =E= 0 ;
A TRANSPORTATION PROBLEM (CANNERY, SEQ=1)
COLUMN LISTING SOLVE CANNERY USING LP FROM LINE 43

---- X
SHIPMENTS (CASES OF TINS PER YEAR)

X(SEATTLE, NEW-YORK)
(.LO, .L, .UP = 0, 0, +INF)
1  SUPPLY(SEATTLE)
1  DEMAND(NEW-YORK)
-25  COST

X(SEATTLE, CHICAGO)
(.LO, .L, .UP = 0, 0, +INF)
1  SUPPLY(SEATTLE)
1  DEMAND(CHICAGO)
-17  COST

X(SEATTLE, KANSAS-CTY)
(.LO, .L, .UP = 0, 0, +INF)
1  SUPPLY(SEATTLE)
1  DEMAND(KANSAS-CTY)
-18  COST

REMAINING 3 ENTRIES SKIPPED

---- TRCOST
TRANSPORT COST (DOLLARS PER YEAR)

TRCOST
(.LO, .L, .UP = -INF, 0, +INF)
1  COST
A TRANSPORTATION PROBLEM (CANNERY, SEQ=1)
MODEL STATISTICS SOLVE CANNERY USING LP FROM LINE 43

MODEL STATISTICS
NUMBER OF MAJOR ROWS = 3
NUMBER OF MINOR ROWS = 6
NUMBER OF MAJOR COLS = 2
NUMBER OF MINOR COLS = 7
NUMBER OF NON-ZEROES = 19
MODEL GENERATION = 0.096 SECONDS
EXECUTION TIME = 0.416 SECONDS
A TRANSPORTATION PROBLEM (CANNERY, SEQ=1)
SOLUTION REPORT   SOLVE CANNERY USING LP FROM LINE 43

SOLVE SUMMARY
MODEL   CANNERY
OBJECTIVE  TRCOST
TYPE     LP
DIRECTION MINIMIZE
SOLVER  MINOS3
FROM LINE 43

**** SOLVER STATUS  1 NORMAL COMPLETION
**** MODEL STATUS   1 OPTIMAL
**** OBJECTIVE VALUE 16800.0000

RESOURCE USAGE, LIMIT  0.060  1000.000
ITERATION COUNT, LIMIT  4       1000

MINOS 3.4/ALTED
B. A. MURTAGH AND M. A. SAUNDERS,
DEPARTMENT OF OPERATIONS RESEARCH,
STANFORD UNIVERSITY,
STANFORD CALIFORNIA  94305  U.S.A.

WORK SPACE NEEDED (ESTIMATE)  --  3978 WORDS.
WORK SPACE AVAILABLE  --  100000 WORDS.
EXIT -- OPTIMAL SOLUTION FOUND.

EQU SUPPLY

AVAILABILITY CONSTRAINT

LOWER  LEVEL  UPPER  MARGINAL
SEATTLE -INF  350.0000  350.0000  EPS
SAN-DIEGO -INF  550.0000  600.0000  

EQU DEMAND

REQUIREMENTS CONSTRAINT

LOWER  LEVEL  UPPER  MARGINAL
NEW-YORK  300.0000  300.0000  +INF  25.0000
CHICAGO  300.0000  300.0000  +INF  17.0000
KANSAS-CTY  300.0000  300.0000  +INF  14.0000

LOWER  LEVEL  UPPER  MARGINAL

EQU COST

COST  COST DEFINITION

1.0000

VAR X

SHIPCMENTS (CASES OF TINS PER YEAR)

LOWER  LEVEL  UPPER  MARGINAL
SEATTLE .NEW-YORK  .  50.0000  +INF  
SEATTLE .CHICAGO  .  300.0000  +INF  
SEATTLE .KANSAS-CTY  .  +INF  4.0000
A TRANSPORTATION PROBLEM (CANNERY, SEQ=1)
SOLUTION REPORT  SOLVE CANNERY USING LP FROM LINE 43

VARIABLES

<table>
<thead>
<tr>
<th>VAR</th>
<th>LOWER</th>
<th>LEVEL</th>
<th>UPPER</th>
<th>MARGINAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHIPMENTS (CASES OF TINS PER YEAR)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAN-DIEGO.NEW-YORK</td>
<td>.</td>
<td>250.0000</td>
<td>+INF</td>
<td>.</td>
</tr>
<tr>
<td>SAN-DIEGO.CHICAGO</td>
<td>.</td>
<td>300.0000</td>
<td>+INF</td>
<td>1.0000</td>
</tr>
<tr>
<td>SAN-DIEGO.KANSAS-CTY</td>
<td>.</td>
<td>.</td>
<td>+INF</td>
<td>.</td>
</tr>
</tbody>
</table>

#TRANSPORT COST (DOLLARS PER YEAR)

<table>
<thead>
<tr>
<th>TRCOST</th>
<th>LOWER</th>
<th>LEVEL</th>
<th>UPPER</th>
<th>MARGINAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-INF</td>
<td>16800.0000</td>
<td>+INF</td>
<td>.</td>
</tr>
</tbody>
</table>

**** REPORT SUMMARY :

0 NONOPT ****
0 INFEASIBLE ****
0 UNBOUNDED ****
0 ERRORS ****
A TRANSPORTATION PROBLEM (CANNERY, SEQ=1)
EXECUTING

EXECUTION TIME = 0.457 SECONDS
FILE: ORIG	 PROB	 A1 VM/SP (CMS) R3.1 6/12/85 PUT8502

MAX 75X1 + 6X2 + 3X3 + 33X4
ST 774X1 + 76X2 + 22X3 + 42X4 <= 875
67X1 + 27X2 + 794X3 + 53X4 <= 875
END
INTEGER X1
INTEGER X2
INTEGER X3
INTEGER X4
GO
MAX $75 \times 1 + 6 \times 2 + 3 \times 3 + 33 \times 4$

LP OPTIMUM FOUND AT STEP 4

OBJECTIVE FUNCTION VALUE
1) $113.837265$

VARIABLE VALUE REDUCED COST
X1 1.000000 -14.228930
X2 0.506042 0.000000
X3 0.933674 0.000000
X4 1.000000 -29.623016

ROW SLACK OR SURPLUS DUAL PRICES
2) 0.000000 0.078377
3) 0.000000 0.001607

NO. ITERATIONS= 4
BRANCHES= 0 DETERM.= $5.975 \times 10^4$

SET X2 TO 1 AT 1 BND= 113.15228 TWIN= 111.68181
SET X1 TO 1 AT 2 BND= 100.64285 TWIN= 111.68181
SET X4 TO 0 AT 3 BND= 83.95087 TWIN= 84.000015
SET X3 TO 0 AT 4 BND= 81.000000 TWIN= 80.999985

NEW INTEGER SOLUTION OF 81.0000 AT BRANCH 4 PIVOT 10

OBJECTIVE FUNCTION VALUE
1) $81.000000$

VARIABLE VALUE REDUCED COST
X1 1.000000 -75.000000
X2 1.000000 -6.000000
X3 0.000000 -3.000000
X4 0.000000 -33.000000

ROW SLACK OR SURPLUS DUAL PRICES
2) 25.000000 0.000000
3) 781.000000 0.000000

NO. ITERATIONS= 10
BRANCHES= 4 DETERM.= $1.000 \times 10^0$

BOUND ON OPTIMUM: 113.8373

DELETE X3 AT LEVEL 4
FLIP X4 TO 1 WITH BOUND 84.00002
DELETE X4 AT LEVEL 3
FLIP X1 TO 0 WITH BOUND 111.6818
DELETE X1 AT LEVEL 2
FLIP X2 TO 0 WITH BOUND 111.6818
SET X3 TO 0 AT 2 BND= 107.99992 TWIN= 107.99986
NEW INTEGER SOLUTION OF 108.000 AT BRANCH 5 PIVOT 14
OBJECTIVE FUNCTION VALUE
1) 108.000000

VARIABLE VALUE REDUCED COST
X1 1.000000 -75.000000
X2 0.000000 -6.000000
X3 0.000000 -3.000000
X4 1.000000 -33.000000

ROW SLACK OR SURPLUS DUAL PRICES
2) 59.000000 0.000000
3) 755.000000 0.000000

NO. ITERATIONS= 14
BRANCHES= 5 DETERM.= 1.000E 0
BOUND ON OPTIMUM: 113.8373
DELETE X3 AT LEVEL 2
DELETE X2 AT LEVEL 1
ENUMERATION COMPLETE. BRANCHES= 5 PIVOTS= 14

LAST INTEGER SOLUTION IS THE BEST FOUND
RE-INSTALLING BEST SOLUTION...
FILE: GAMRPT  EXEC   A1 VM/SP (CMS) R3.1 6/12/85 PUT8502

FORTVS TOGAMRPT (LANGLVL(77) FLAG(E)
EXEC GLOB FORTRAN
LOAD TOGAMRPT
GENMOD TOGAMRPT
Originating Linkid: SMUVM1
Originating Userid: E4M10014
Distribution Code: BIN695
Spool file number: 8903
File Size - Recs: 00000007
File name and type: TOMPSX EXEC
Origin Time/Date: 5/13/86 4:32:35 C.D.T.
Originating Linkid: SMUVM1
Originating Userid: E4M10014
Distribution Code: BIN695
Spool file number: 8906
File Size - Recs: 00000007
File name and type: POSTOPT EXEC
Origin Time/Date: 5/13/86 4:32:36 C.D.T.
FILE: POSTOPT EXEC A1 VM/SP (CMS) R3.1 6/12/85 PUT8502

&CONTROL NOMSG OFF
F1 5 DISK FILE DPLUSER1 A (RECFM VBS LRECL 204 BLKSIZE 1024
F1 6 DISK GAMSLPF FLPSOL A
F1 7 DISK GAMSLPF FLPSAT A
TOGAMRPT
ERASE FILE DPLUSER1 A
FILE: MPS EXEC A1 VM/SP (CMS) R3.1 6/12/85 PUT8502

&CONTROL OFF
&IF INDEX = 1 &GOTO -EXEC
FI SYSIN DISK &1 MPSCL * (RECFM F BLOCK 80
&TYPE DPLCOMP
DPLCOMP
&IF &RETCODE = 0 &GOTO -CONT
&TYPE COMPILE ERROR
-EDG &EXIT
-CONT &IF &2 = NOEX &GOTO -EDG
&GOTO -CP
-EXEC &2 = &1
-CP FI SYSPRINT DISK (DISP MOD
FI SYSIN DISK &2 DATA * (RECFM F BLOCK 80
FI DPLUSER1 DISK (RECFM VBS LRECL 204 BLKSIZE 1024
DPLEXEC
FI SYSPRINT CLEAR
&TYPE RETCODE = &RETCODE
&EXIT
&CONTROL NOMSG OFF
CLRSCRN
&BEGTYPE

*****************************************************************************
  *  *  
  *  *  
  *MPSX370/MIP  *  
  *  *  
  *****************************************************************************

... PREPROCESSING STARTS TO WORK...

&END
EXEC TOMPSX
&BEGTYPE

*****************************************************************************
  *  *  
  *  *  
  *MPSX370/MIP  *  
  *  *  
  *****************************************************************************

... IBM/MPSX370 OPTIMIZER IS WORKING...

&END
EXEC MPSX MPS TOMPSX TOMPSX
&IF &RETCODE NE 0 &GLOBAL3 = 100
CLRSCRN
&BEGTYPE

*****************************************************************************
  *  *  
  *  *  
  *MPSX370/MIP  *  
  *  *  
  *****************************************************************************

... POST PROCESSING STARTS TO WORK...

&END
EXEC POSTOPT
ERASE FILE SCRATCH1 A
ERASE FILE SCRATCH2 A
ERASE FILE SCRATCH3 A
ERASE FILE SCRATCH4 A
ERASE FILE PROBFILE A
ERASE FILE SYSMLCP A
&CONTROL OFF NOMSG
&IF &INDEX < 2 &GOTO -EXPLAIN
&GOTO -START
-EXPLAIN CLRSCRN
&BEGTYPE

FILE: MPSX EXEC A2 VM/SP (CMS) R3.1 6/12/85 PUT8502

*PROCEDURE NAME - MPS
*FUNCTION - ACCESS AND INVOKE MPSX SYSTEM.
*COMMAND - MPSX <MODE> <MPSXCL> <MPSXDATA>
*ENTRY CONDITIONS -
*<MODE> - MPS - FOR EXECUTION OF MPS CONTROL LANGUAGE
*ECL - FOR COMPILATION AND EXECUTION OF
*EXTENDED CONTROL LANGUAGE.
*<MPSXCL> - CONTAINS MPSX PROGRAM STATEMENTS.
*OR IF THIS IS THE LAST PARAMETER, THIS
*NAME IS ASSUMED TO BE A DATA FILE, AND
*THAT THE PROBLEM PROGRAM HAS ALREADY BEEN
*COMPILED. THIS WILL EXECUTE ONLY AND NOT
*RECOMPILE THE PROGRAM.
*<MPSXDATA> - CONTAINS THE MPSX DATA INPUT.
*THIS PARAMETER MAY BE SPECIFIED AS
'NOEX' TO SUPPRESS PROGRAM EXECUTION

&END
&EXIT

-START
--------SAVE PARAMETERS
&PL1 =
&P1 = &1
&P2 = &2
&P3 = &3
STATE PLIOPT MODULE *
&IF &RETCODE = 0 &IF &P1 = ECL &PL1 = PL1
*--------TEST FOR DISK ALREADY LINKED
STATE DPLCOMP MODULE *
&IF &RETCODE = 0 &GOTO -LINKED
*--------LINK MPSX DISK
EXEC $$$$SETUP MPSX &PL1
  "-LINKED
  &STACK RT
  &IF .&P1 = .MPS &GOTO -&P1
  &IF .&P1 = .ECL &GOTO -&P1
  EXEC SETUP RESET
  &GOTO -EXPLAIN
  "
  -MPS EXEC MPS &P2 &P3
  &RCD = &RETCODE
  &GOTO -EXIT
  "
  -ECL EXEC ECL &P2 &P3
  &RCD = &RETCODE
  "
  -EXIT
  EXEC $$$$SETUP RESET
  &EXIT &RCD
  "
  -ERROR1 &TYPE NOT ABLE TO LINK TO MPSX DISK
  &TYPE CONTACT ENG. & SCIENTIFIC SOFTWARE GROUP 471-3241
  &EXIT 999
  "
DEF TITLE
GENERAL ALGEBRAIC MODELING SYSTEM
RESWORDS

EQ
GT
GE
LT
LE
NE
OR
NO
NA
XOR
YES
INF
EPS
SUM
SET
ORD
AND
NOT
ALL
PROD
SMIN
SMAX
SETS
FREE
CARD
LOOP
SOS1
SOS2
MODEL
ABORT
TABLE
ALIAS
SOLVE
USING
ASSIGN
BINARY
SCALAR
MODELS
SYSTEM
OPTION
INTEGER
SCALARS
OPTIONS
DISPLAY
ACRONYM
ACRONYMS
VARIABLE
POSITIVE
NEGATIVE
EQUATION
PARAMETER
VARIABLES
FILE: COMP
DATA A2 VM/SP (CMS) R3.1 6/12/85 PUT8502

EQUATIONS
PARAMETERS
MINIMIZING
MAXIMIZING
PROCNAMES
LP
MIP
RMIP
NLP
TV
SAMBAL
MATBAL
DNLP

ALGNAMES
MINOS3 1 LP RMIP
MPSX 1 LP MIP RMIP
APEX3 0 LP MIP RMIP
APEX4 0 LP MIP RMIP
MINOS5 1 LP RMIP NLP DNLP
NPSOL 0 LP RMIP NLP DNLP
GRG2 0 NLP DNLP
CONOPT 0 NLP DNLP

SAMLIB 1 TV MATBAL SAMBAL

TVLABELS

TVACRONYM

GROUP
MF
INST
AC
2
SPECS
VSHR
CD
2
FIX
PQ

(* READ IN, NOT USED *)
FILE: COMP	DATA A2 VM/SP (CMS) R3.1 6/12/85 PUT8502

SDOUT  2 3 0
LIMCOL  2 4 3
LIMROW  2 5 3
WORK   2 6 0
DOMLIM 2 7 0
DECIMALS 2 8 3
INTEGER1 2 9 0
INTEGER2 2 10 0
INTEGER3 2 11 0
INTEGER4 2 12 0
INTEGER5 2 13 0
RESLIM 3 0 1000.0
PERT  3 1 0.0
OPTCR  3 2 0.1
OPTCA  3 3 0.0
BRATIO 3 4 0.25
REAL1  3 5 0.0
REAL2  3 6 0.0
REAL3  3 7 0.0
REAL4  3 8 0.0
REAL5  3 9 0.0
SOLVEOPT 4 0 MERGE REPLACE
SYSOUT 4 1 OFF ON
ERRORMSGS
0 INTEGER NUMBER EXPECTED
1 REAL NUMBER EXPECTED
2 IDENTIFIER EXPECTED
3 ELEMENT EXPECTED
4 '+' OPERATOR EXPECTED
5 '*' OPERATOR EXPECTED
6 '**' OPERATOR EXPECTED
7 '(' EXPECTED
8 ')' EXPECTED
9 '$' OPERATOR EXPECTED
10 '::' EXPECTED
11 '::' EXPECTED
12 '::' EXPECTED
13 '+' OPERATOR EXPECTED
14 'LT' OPERATOR EXPECTED
15 'OR' OPERATOR EXPECTED
16 'INF' EXPECTED
17 'SUM' OPERATOR EXPECTED
18 'PARAMETER' EXPECTED
19 'AND' EXPECTED
20 'NOT' OPERATOR EXPECTED
21 'ALL' EXPECTED
22 'POSITIVE' EXPECTED
23 'LOOP' EXPECTED
24 'TABLE' EXPECTED
25 'ALIAS' EXPECTED
26 'SOLVE' EXPECTED
27 'USING' EXPECTED
28 'OPTION' EXPECTED
29 'DISPLAY' EXPECTED
30 'ORD' EXPECTED
'CARD' EXPECTED
'MODEL' EXPECTED
'MAXIMIZING' EXPECTED
'=' OPERATOR EXPECTED
'='=' OPERATOR EXPECTED
QUOTE EXPECTED
NON BLANK CHARACTER(S) BEYOND COLUMN 120
ILLEGAL CHARACTER
TOO MANY DIGITS IN INTEGER
TOO MANY DIGITS IN REAL
EXPONENT EXPECTED
EXPONENT OUT OF RANGE
NUMBER OUT OF RANGE, SET TO UNDEFINED
SYMBOL TOO LONG
IDENTIFIER TOO LONG
ELEMENT TOO LONG
UNKNOWN SUFFIX IGNORED
CLOSING QUOTE MISSING IN ELEMENT
ELEMENT IS EMPTY
DOMAIN CHECKING SUPPRESSED FOR THIS SYMBOL
UNRECOGNIZED STATEMENT
NUMBER (PRIMARY) EXPECTED
UNKNOWN IDENTIFIER ENTERED AS SET
SET EXPECTED
ONE DIMENSIONAL SET EXPECTED
SET IS EMPTY
SUFFIX FOR A SET NOT ALLOWED
SET IS UNDER CONTROL ALREADY
SET IDENTIFIER OR '(' EXPECTED
BASE OF POWER NOT A REAL NUMBER
EXPONENT OF POWER NOT A REAL NUMBER
DIVISION NOT DEFINED FOR A SET
INCOMPATIBLE OPERANDS FOR MULIPLICATION
INCOMPATIBLE OPERANDS FOR ADDITION
INCOMPATIBLE OPERANDS FOR RELATION
UNKNOWN SYMBOL, ENTERED AS PARAMETER
SYMBOL HAS NO DATA
ILLEGAL SUFFIX IGNORED
MISSING SUFFIX
ILLEGAL IDENTIFIER (MODEL OR FUNCTION)
SET IDENTIFIER EXPECTED
MAXIMUM NUMBER OF DIMENSIONS EXCEEDED
REAL VALUE FOR LAGOPERATOR EXPECTED
DIMENSION DIFFERENT
UNCONTROLLED SET ENTERED AS CONSTANT
EQUATION REDEFINED
SYMBOL AND RIGHT SIDE OF DIFFERENT TYPE IN ASSIGNMENT
EQUATION EXPECTED
UNCONTROLLED SET(S) IN LAG-EXPRESSION
SET FOR 'ORD' IS NOT CONTROLLED
SET FOR 'ORD' HAS BEEN MODIFIED
MORE THAN ONE KNOWN SET IN ALIAS LIST
NO KNOWN SET FOUND IN ALIAS LIST
NO SETS FOUND TO BE ALIASED
AN ALIASED SET IS NOT ALLOWED HERE
160 UNIQUE ELEMENT EXPECTED
161 CONFLICTING DIMENSIONS IN ELEMENT
162 MACRO WITHOUT NUMERIC PART
163 FIRST PART OF MACROS NOT IDENTICAL
164 SECOND PART OF MACROS NOT IDENTICAL
165 FIRST NUMBER GREATER THAN SECOND NUMBER IN MACRO
166 '=' ASSIGNMENT EXPECTED
167 REDEFINITION OF LOOP CONTROLLED SET
168 ASSIGNMENT NOT ALLOWED
169 EQUATION WITHIN A LOOP IS NOT ALLOWED
170 DOMAIN VIOLATION FOR ELEMENT
171 DOMAIN VIOLATION FOR SET
172 ELEMENT IS REDEFINED
173 DISPLAY WITHIN A LOOP NOT ALLOWED (PC ONLY)
180 'VARIABLE' EXPECTED
181 CLOSING '/' MISSING
184 DOMAIN LIST REDEFINED
185 SET IDENTIFIER OR '#' EXPECTED
186 ILLEGAL DATALIST IGNORED
187 ASSIGNED SET USED AS DOMAIN
188 ASSIGNING TO SET USED AS DOMAIN
189 DOMAINTLIST NOT ALLOWED FOR SCALAR
190 TEXT TOO LONG, TRUNCATED
191 CLOSING QUOTE MISSING
192 IDENTIFIER OR QUOTED STRING EXPECTED
193 TEXT FOR SYMBOL REDEFINED
194 SYMBOL REDEFINED
195 SYMBOL REDEFINED WITH A DIFFERENT TYPE
196 VARIABLE REDEFINED WITH A DIFFERENT SUBTYPE
200 FUNCTION NOT ALLOWED HERE
201 INVALID ARGUMENT FOR FUNCTION
202 TOO MANY ARGUMENTS FOR FUNCTION
203 TOO FEW ARGUMENTS FOR FUNCTION
210 UNKNOWN DOLLAR CONTROL OPTION, VALID ARE -
211 ONMARGIN OFFMARGIN SINGLE DOUBLE
212 ONDOLLAR OFFDOLLAR EJECT
213 ONSYMLIST OFFSYM LIST ONUELLIST OFFUELLIST
214 ONTEXT OFFTEXT (BRACKET A BLOCK OF COMMENT LINES)
215 LINES (FOLLOWED BY LINE COUNT)
216 TITLE STITLE (FOLLOWED BY NEW TITLE/SUBTITLE)
217 MINCOL MAXCOL (FOLLOWED BY COLUMN POSITION)
218 DOLLAR COMMENT (FOLLOWED BY 1 CHARACTER)
219 HIDDEN (FOLLOWED BY UNLISTED COMMENT)
220 LEFT MARGIN MUST BE LESS THAN RIGHT MARGIN
221 RIGHT MARGIN MUST BE GREATER THAN LEFT MARGIN AND 15
222 RIGHT MARGIN MUST BE LESS THAN 121
223 UNKNOWN DOLLAR DEBUG OPTION, VALID ARE -
224 DMPSYM DMPCPL DMPTPL DMPLINES MEASURE
225 NO PREVIOUS "ONTENT"
226 "OFFTEXT" EXPECTED
227 DOLLAR AND COMMENT CHARACTERS MUST BE DIFFERENT
228 SINGLE CHARACTER BETWEEN BLANKS EXPECTED
229 PARTIAL OVERLAPPING COLUMN ADJUSTED
FILE: COMP  DATA  A2  VM/SP (CMS) R3.1 6/12/85 PUT8502

221 TOTAL OVERLAPPING COLUMN IGNORED
222 COLUMN HAS DIFFERENT DIMENSION
223 OVERLAPPING ROWNAME, ADJUSTED
224 OVERLAPPING ENTRY IGNORED
225 FLOATING ENTRY IGNORED
226 MULTIPLE ENTRY IGNORED
227 '/' IN MODEL LIST EXPECTED
230 IDENTIFIER IS NOT A MODEL
231 MODEL REDEFINED X
232 UNDEFINED IDENTIFIER(S) IN MODEL LIST
233 EMPTY MODEL LIST
234 ILLEGAL IDENTIFIER IN MODEL LIST
235 TYPE OUT OF SEQUENCE
236 DUPLICATE IDENTIFIER(S) IN MODEL DEFINITION
240 UNKNOWN IDENTIFIER, ENTERED AS BEST GUESS
241 MODEL HAS NOT BEEN DEFINED
242 UNKNOWN IDENTIFIER, ENTERED AS VARIABLE X
243 OBJECTIVE IS NOT A VARIABLE
244 UNKNOWN PROCESSOR NAME
245 OBJECTIVE VARIABLE NOT REFERENCED IN MODEL
246 OBJECTIVE VARIABLE IS NOT A FREE VARIABLE
247 NO ALGORITHM AVAILABLE FOR THIS CLASS OF MODEL
250 UNKNOWN OPTION. VALID ARE -
250 INTEGER1 TO INTEGER5
250 LIMROW LIMCOL DOMLIM ITERLIM WORK (INTEGER VALUES)
250 DECIMALS (0..8)
250 OPTCA OPCR
250 REAL1 TO REAL5 RESLIM BRATIO (REAL VALUES)
250 MEASURE EJECT (NO VALUES)
250 SYSOUT (ON/OFF) SOLVOPT (MERGE/REPLACE)
251 UNKNOWN OPTION VALUE
252 SYSTEM OPTION VALUE OUTSIDE RANGE 0..32767
253 UNKNOWN ALGORITHM
254 ALGORITHM NOT LICENSED
255 ALGORITHM NOT SUITABLE FOR PROCESS
256 ERROR(S) IN ANALYZING SOLVE STATEMENT
257 SOLVE STATEMENT NOT CHECKED BECAUSE OF PREVIOUS ERRORS
258 OPTION "DECIMALS" MUST BE IN RANGE 0..8
260 INSTRUCTION SPACE OVERFLOW
261 CONSTANT POOL OVERFLOW
262 ILLEGAL SYNTAX FOR TV SOLVE
265 EQUATION HAS NOT BEEN DEFINED
266 IDENTIFIER IS NOT AN EQUATION
267 FEATURE NOT IMPLEMENTED
269 A TV MODEL MUST CONTAIN 3 IDENTIFIERS
299 UNEXPECTED END OF FILE
300 REMAINING ERRORS NOT PRINTED FOR THIS LINE
999 *** SENTINEL ***
USER PROCEDURE TO CONVERT THE SOLUTION FILE FROM MPSX/370 TO BE THE FILES COMPATIBLE TO CAMS

INTEGER*4 FILE, LIST, RPT, NOCOL, NOCOL2, I, J, L, M, N, P
INTEGER*4 TYPE(60), VALNUM(50)
REAL*8 VOBJ
REAL*8 NOITER, NOROW, NOREC, NOHEAD, NDUM
REAL*8 COLUMN(30), VALUES(25)
REAL*8 NAME, ENDATA, ENDSEC
REAL*8 BASIS, BSLO, BSHI, EQUAL
REAL*4 VALALF(50)

DATA ENDATA/'ENDATA /
DATA OBJ /'OBJ /
DATA FUNCTION/'FUNCTION /
DATA ITER/'ITERN /
DATA ROWS/'ROWS /
DATA COLM/'COLUMNS /
DATA TARG/'TARGET /
DATA STAT/'STATUS /
DATA METH/'METHOD /
DATA BASIS/'BS /
DATA EQUAL/'EQ /
DATA OPTM/'OPTM /
DATA INTG/'ITGR /
DATA INFES/'NFES /
DATA NOPT/'NOPT /

EQUIVALENCE (VALUES(1), VALNUM(1), VALALF(1))

FILE = 5
LIST = 6
RPT = 7

SKIP THE NAME,XDATA RECORD

READ(FILE)

READ(FILE) NAME, NOCOL
NOCOL2 = 2 * NOCOL
READ(FILE) (COLUMN(N), N=1, NOCOL)
READ(FILE) (TYPE(N), N=1, NOCOL2)
J = 0
DO 2 I=2, NOCOL2, 2
   J = J + TYPE(I)
2 CONTINUE
J = J/4
READ(FILE) (VALALF(N), N=1, J)

PRINT THE IDENTIFICATION ARRAY, ONE VALUE PER LINE

J = 0
FILE: TOGMRPT FORTRAN A1 VM/SP (CMS) R3.1 6/12/85 PUT8502

DO 20 N=1,NOCOL
L = J/4 + 1
M = L + 1
P = L + 19
IF(TYPE(2*N-1) - 2) 10,14,12
C
C NUMERIC - INTEGER - VALUE
C
10 IF(COLUMN(N).NE.ITER) GO TO 33
    NOITER = VALNUM(L)
    GO TO 19
33 IF(COLUMN(N).NE.ROWS) GO TO 44
    NOROW = VALNUM(L)
    GO TO 19
44 IF(COLUMN(N).NE.COLM) GO TO 19
    NOCOLS = VALNUM(L)
    GO TO 19
C
C NUMERIC - REAL - VALUE
C
12 IF(COLUMN(N).NE.FUNC) GO TO 19
    VOBJ = VALALF(L)
    GO TO 19
C
C ALPHAMERIC VALUE, LENGTH MAY BE 4, 8, OR 80
C
14 IF(TYPE(2*N) - 8) 15,17,18
C
C ALPHAMERIC VALUE - LENGTH = 4
C
15 IF(COLUMN(N).NE.STAT) GO TO 55
    ISTAT = VALALF(L)
    GO TO 19
55 IF(COLUMN(N).NE.TARG) GO TO 66
    ITARG = VALALF(L)
    GO TO 19
66 IF(COLUMN(N).NE.METH) GO TO 19
    IMETH = VALALF(L)
    GO TO 19
C
C ALPHAMERIC VALUE - LENGTH = 8
C
17 GO TO 19
C
C ALPHAMERIC VALUE - LENGTH = 80
C
18 CONTINUE
C
19 J = J + TYPE(2*N)
20 CONTINUE
C
C WRITE OUT THE STATUS FILE FOR GAMS REPORT
C
WRITE(RPT,1001)
1001 FORMAT('1 ',/,'EXECUTOR. IBM MPSX/370-MIP/370',/) WRITE(RPT,1002) TOG01110
1002 FORMAT(1X,'INTERFACE BETWEEN MPSX/370 AND GAMS IS PROVIDED BY:', TOG01120
       1 /,'CENTER FOR BUSINESS DECISION ANALYSIS', TOG01130
       2 /,'UNIVERSITY OF TEXAS AT AUSTIN', TOG01140
       3 /,'AUSTIN, TEXAS 78712',/) TOG01150
       IF(ISTAT.EQ.OPTM) THEN WRITE(RPT,1003) TOG01160
       1003 FORMAT(1X,'EXIT -- OPTIMAL SOLUTION FOUND.',/) TOG01170
       ELSE IF(ISTAT.EQ.NOPT) THEN WRITE(RPT,1004) TOG01180
       1004 FORMAT(1X,'EXIT -- NON-OPTIMAL BUT FEASIBLE SOLUTION FOUND.',/) TOG01190
       ELSE WRITE(RPT,1005) TOG01200
       1005 FORMAT(1X,'EXIT -- INFEASIBLE SOLUTION FOUND.',/) TOG01210
       ENDIF TOG01220

READ( FILE)
SKIP THE $ENDSEC$ OF THE IDENTIFICATION ARRAY
GET THE ROW AND COLUMN SECTIONS

IF(ISTAT.EQ.OPTM) THEN MSTAT = 1
ELSE IF(ISTAT.EQ.INFES) THEN MSTAT = 4
ENDIF TOG01430

NOREC = NOROW + NOCOLS + 6
NOHEAD = 4.0
NORMAL = 1
WRITE(LIST,2001) NOREC, NOHEAD, NORMAL, MSTAT
RESUSE = 0.0
IKEY1 = 1
IKEY2 = 2
WRITE(LIST,2001) VOBJ, RESUSE, IKEY1, IKEY2
NDUM = 1.0
IKEY3 = 3
IKEY4 = 0
WRITE(LIST,2001) NOITER, NDUM, IKEY3, IKEY4
C IFLAG = -1
21 READ( FILE) NAME, NOCOL
IF(NAME.EQ.ENDATA) GO TO 32

C
READ(FILE)
READ(FILE)

24 READ(FILE) VALUES(N),N=1,NOCOL
IF(VALUES(1).EQ.ENDSEC) THEN
WRITE(LIST,2002)
IFLAG = -1 * IFLAG
GO TO 21
ENDIF

C
INDBS = 0
IF(VALUES(7).EQ.BASIS) THEN
INDBS = 2
ELSE
IF(VALUES(7).EQ.BSHI) THEN
INDBS = 1
ENDIF
ENDIF

C
VALUES(5) = IFLAG * VALUES(5)

25 WRITE(LIST,2001) VALUES(1),VALUES(5), INDBS, IRCST
GO TO 24

C
31 WRITE(LIST,2002)

32 STOP

2001 FORMAT(2(1PE20.11),214)
2002 FORMAT(2(3X,'3.000000031728E+70'),2(3X,'0'))
END
I(* FOR OTHER MACHINES MAY NEED TO ALTER RESET AND REWRITE STATEMENTS *)
I(* DUMMY CLOSE ROUTINE IN THIS FILE MAY NEED DELETING *)
I(* CHECK ALL MIXED INTEGER CONSTRUCTS, ESP SOS VARIABLES AND BOUNDS *)
I(* ALL CONTROL FILE AND FIELD LENGTH STUFF ARE CYBER/APEX SPECIFIC *)
I(* MAKE SURE ALL ERRORS ARE APPROPRIATE *)
I(* CHECK ALL MIXED INTEGER CONSTRUCTS, ESP SOS VARIABLES AND BOUNDS *)
I(* MAKE SURE ALL ERRORS ARE APPROPRIATE *)

PROGRAM GAMS2MIP(FCIN,FDIN, (*) CONTROL, DATA IN *)
FSTAOUT,FSOLOUT, (*) STATUS, SOLUTION OUT IF ERRORS *)
FCOUT,FMOUT,FBOUT,(* CONTROL, MPS, BASIS OUT *)
INPUT,OUTPUT);
ICONST
EFORMATOK = FALSE; (* IF E- FORMAT TOLERATED BY TARGET SOLVER *)
BNDNAME = 'GBND	 ';
RUSNAME = 'GRHS	 ';
MAXRC = 32767;
MOSTA = 13;
SOSTA = 8;
SVAL 1.0;
ITYPE
POSINT = 0..MAXINT;
RCSEQT = 0..MAXRC;
FWID = 0..12;
TMIP = 0..10;
ERRTXT = PACKED ARRAY '1.50' OF CHAR;
WHERE = (GLOBAL,COLUMN,ROW);
SIJBWHERE = (SROW,SCOLUMN,SBOUND,SRHS);
TRHSPTR = @TRHS;
TRHS = RECORD
RHSVAL : REAL;
ROWNUM : RCSEQT;
ROWNXT : TRHSPTR
END;
TBNDPTR @TBND;
TBND = RECORD
BLO, BLVL, BUP :
NDIG : FWID;
COLNUM : RCSEQT;
MIPTYP : TMIP;
BNDNXT : TBNDPTR
END;
TBASPTR = @TBAS;
TBAS = RECORD
INDEX : RCSEQT;
UP : BOOLEAN;
SEQ NO

* HEAD OF LIST FOR RHS *
HRHSPT: TRHSPT;

* HEAD OF LIST FOR BOUNDS *
HBNDPTR: TBNDPTR;

* HEAD OF LIST FOR NON-BASIC ROWS *
HRBASPTR: TBASPTR;

FCOUT, (* CONTROL FILE OUT FOR SOLVER *)
FDIN, (* DATA FILE IN FROM GAMS *)
FCIN, (* CONTROL FILE IN FROM GAMS *)
FSTAOUT, (* STATUS FILE OUT IF ERROR *)
FSOLOUT, (* SOLUTION FILE OUT IF ERROR *)
FMOUT, (* MPS FILE OUT FOR SOLVER *)
FBOUT (* BASIS FILE OUT FOR SOLVER *)

: TEXT;

RPOWEROF10: ARRAY[1..9] OF REAL;
IPPOWEROF10: ARRAY[1..1] OF INTEGER;
WORK, (* IF NON-ZERO USER'S REQUEST FOR MEMORY *)
OBJROSEQ, (* SEQUENCE NUMBER OF OBJECTIVE ROW *)
ITERLIM, (* ITERATION LIMIT *)
DOMLIM, (* EVALUATION ERROR LIMIT NLP ONLY *)
NOTUSED,*
NNDROW, (* COUNT OF NON-BASIC ROWS *)
NUMROWS, (* COUNT OF ROWS *)
NUMCOLS, (* COUNT OF COLUMNS *)
NUMNZ, (* COUNT OF NON-ZEROES IN MATRIX *)
INTEGER;

ENVIRON : WHERE;
SENVIRON : SUBWHERE;
GLOBALEQ: RCSEQT;
ERRMRR: ARRAY[WHERE] OF CHAR;
USER: ARRAY[1..5] OF INTEGER;
XUSER: ARRAY[1..5] OF REAL;
FINF, (* GAMS REPRESENTATION OF +INF *)
MINF, (* -INF *)
RESLIM, (* RESOURCE LIMIT *)
BRATIO (* RATIO NON-B ROWS/B COLUMNS FOR ACCEPTING USER BASIS *)
: REAL;

TYPBIN, (* INDICATORS FOR DISCONTINUOUS VARIABLE TYPES *)
TYPINT, *
TYPPOS1, *
TYPPOS2 : TMIP;
SYSOUT, (* IF T ARRANGE TO COPY SOLVER STATUS FILE TO USER'S *)
DOMIP, (* IF T PROBLEM HAS DISCONTINUOUS VARIABLES *)
ERRORFREE, (* IF T PUT TOGETHER A BASIS AND TELL SOLVER NOT TO *)
NEEDBAS: BOOLEAN;
IDIR : 0..1;
PROCEDURE CLOSE(VAR F : TEXT);
BEGIN
END;
(* DUMMY CLOSE FOR CYBER *)

PROCEDURE PPERROR(ERRNUM : INTEGER;
BEGIN
IF ERRFREE THEN
IF ENVIRON = GLOBAL THEN
BEGIN
WRITELN(FSTAOUT,'=',ERRMKR'ENVIROW');
WRITELN(FSTAOUT,'**** ',TXT)
END
ELSE
WRITE (FSTAOUT,'=',ERRMKR'ENVIRON');
WRITE (' ',GLOBALSEQ:1,' ',ERRNUM:1,' ',TXT);  
END;
(* WRITE A RECORD FOR THIS ERROR *)

FUNCTION PRDIGITS(K : POSINT) : FWID;
VAR
G : POSINT;
BEGIN
IF K >= IPOWEROF10^"4" THEN
G := 4
ELSE
BEGIN
G := 1;
WHILE K >= IPOWEROF10^"G" DO G := G+1
END;
PRDIGITS := G
END; (* PRDIGITS *)
PROCEDURE COEFFOUT(X : REAL; VAR F : TEXT);
(* WRITE A COEFFICIENT IN FIXED FIELD WIDTH *)
VAR
Y : REAL;
N : INTEGER;
begin
  i := 0..12;
  begin
    y := abs(x);
    if y > rpowerof10 \(-9\)
      then
        begin
          write(f,x:12);
          if not eformatok
            then
              case senvironment of
                srow:
                  ;
                scolumn:
                  perror(1,'coefficient format overflow');
                srhs:
                  perror(2,'rhs format overflow');
                sbound:
                  perror(3,'bound format overflow');
                end;
          end;
        else
          i := 1;
          while y >= rpowerof10 \(-12\)
            do
              i := i + 1;
              write(f,x:12:10-i);
            end;
      end;
    end;
  end;
end;
(* coeffout *)

procedure initialize;
var
  comp,
  c : integer;
begin
  rpowerof10 \("\) := 9.999;
  for c := 2 to 9
    do rpowerof10 \("c\) := 10.0*rpowerof10 \("c-1\)";
  ipowerof10 \("1\) := 10;
  (* 100 000 is not a good integer on some machines *)
  for c := 2 to 4
    do ipowerof10 \("c\) := 10*ipowerof10 \("c-1\)";
  errorfree := true;
  environ := global;
  errmkrglobal := '1'; errmkrrow := '5'; errmkrcolumn := '6';
  dtxt 0 := 'min'; dtxt 1 := 'max';
  reset(fdin);
  reset(fcin);
  rewrite(fmout);
  rewrite(fbout);
  readln(fcin,notused,comp,c,typbin,typint,typsos1,typsos2);
  domip := c = comp;
  readln(fcin,numrows);
  readln(fcin,numcols);
  readln(fcin,numnnz);
  readln(fcin,numused);
  readln(fcin,objroseq);
  readln(fcin,idir);
READLN(FCIN, RESLIM);
FOR C := 1 TO 5
DO READLN(FCIN, WORK, C); SYSOUT := C = 1;
CLOSE(FCIN);
READLN(FCIN, BRATIO);
FOR C := 1 TO 5
DO READLN(FCIN, USER"C");
FOR C := 1 TO 5
DO READLN(FCIN, USER"C");
READLN(FCIN, USER"C");
READLN(FCIN, DOMLIM);
READLN(FCIN, ITERLIM);
READLN(FCIN, RESLIM);
IF NUMROWS > MAXRC THEN PPERROR(0,'TOO MANY ROWS');
IF NUMCOLS > MAXRC THEN PPERROR(0,'TOO MANY COLUMNS');
PROCEDURE ELIMBLIST(VAR BPTR : TBASPTR);
(* GET RID OF THE BASIS LIST *)
VAR
  B1PTR,
  B2PTR : TBASPTR;
BEGIN
  BPTR := NIL
END; (* ELIMBLIST *)
PROCEDURE PROCESSROWS;
VAR
  RST : ARRAY[0..3] OF CHAR;
  NRBPTR,
  XRBPTR : TBASPTR;
  XRHS PTR,
  NRHS PTR : TRHS PTR;
  J : RCSEQT;
  MARGNFLAG,
  ISTYP : INTEGER;
  VALU : REAL;
BEGIN
  HRHS PTR := NIL;
  HRBAS PTR := NIL;
  NNBROW := 0;
READLN(FDIN);
WRITELN(FMOUT,'NAME':10,'GAMS');
ENVIRON := ROW;
SENVIRON := SROW;
FOR J := 1 TO NUMROWS
DO BEGIN
GLOBALSEQ : J;
READLN(FDIN,ITYP,VALU,MARGNFLAG);
IF VALU <> 0.0
THEN
BEGIN
NEW(NRHSPTR);
WITH NRHSPTR@ DO BEGIN
ROWNUM : J;
RHSVAL : VALU;
ROWNXT := NIL;
END;
IF HRHSPTR = NIL
THEN
HRHSPTR := NRHSPTR
ELSE
XRHSPTR@.ROWNXT := NRHSPTR;
END;
IF MARGNFLAG = 1
THEN
BEGIN
NEW(NRBPTR);
WITH NRBPTR@ DO BEGIN
INDX := J;
(* BINDING G CONSTRAINTS --> SLACK AT UPPER *)
UP := ITYP = 1;
INDXXNT := NIL;
END;
IF HRBASPTR = NIL
THEN
HRBASPTR := NRBPTR
ELSE
XRBPTR@.INDXXNT := NRBPTR;
END;
IF HRBASPTR = NIL
THEN
HRBASPTR := NRBPTR
ELSE
XRBPTR@.INDXXNT := NRBPTR;
END;
(* SAVE AWAY INDEX OF NON-BASIC ROW *)
BEGIN
NEW(NRBPTR);
WITH NRBPTR@ DO BEGIN
INDX := J;
(* BINDING G CONSTRAINTS --> SLACK AT UPPER *)
UP := ITYP = 1;
INDXXNT := NIL;
END;
IF HRBASPTR = NIL
THEN
HRBASPTR := NRBPTR
ELSE
XRBPTR@.INDXXNT := NRBPTR;
END;
IF HRBASPTR = NIL
THEN
HRBASPTR := NRBPTR
ELSE
XRBPTR@.INDXXNT := NRBPTR;
END;
(* SAVE AWAY IN RHS LIST *)
BEGIN
NEW(NRHSPTR);
WITH NRHSPTR@ DO BEGIN
HRHSPTR := NRHSPTR
END;
IF HRBASPTR = NIL
THEN
HRBASPTR := NRBPTR
ELSE
XRBPTR@.INDXXNT := NRBPTR;
END;
IF HRBASPTR = NIL
THEN
HRBASPTR := NRBPTR
ELSE
XRBPTR@.INDXXNT := NRBPTR;
END;
(* PROCESSROWS *)
PROCEDURE PROCESSCOLS;
VAR COEFF,
LOWER,
LEVEL,
UPPER : REAL;
PRR,
PRC : FWID;
SAVTYP,
DCVTYP : TMIP;
SAVINDX,
SOSINDX,
MARGNFLAG,
NNZ,
ROW,
K : POSINT;
SAVCOL,
J : RCSEQT;
WBNDPTR : TBNDPTR;
XRASPTR : TBASPTR;
DOSOS,
NEEBNDREC : BOOLEAN;
PROCEDURE SAVBNDREC(COL : RCSEQT);
VAR NBNDPTR :
BEGIN
NEW(NBNDPTR);
WITH NBNDPTR AT
DO BEGIN
MDIG := PRC;
BLO := LOWER;
BLVL := LEVEL;
BUP := UPPER;
COLNUM := COL;
MPTYPT := DCVTYP;
BNDXT := NIL;
END;
IF HBNDPTR = NIL
THEN HBNDPTR := NBNDPTR
ELSE WBNNDPTR.BNDXT := NBNDPTR;
WBNDPTR := NBNDPTR;
END;
FUNCTION TERMSOS(COL : RCSEQT) : BOOLEAN;
(* WRITES BOUND RECORD TO TERMINATE SOS SET. LO LVL UP TYP ETC INCOR *)
(* BUT WILL BE IGNORED *)
VAR NUM : RCSEQT;
BEGIN
### CROSS REFERENCE LISTING

<table>
<thead>
<tr>
<th>IDENTIFIER</th>
<th>Definition</th>
<th>Attributes</th>
<th>Format: &lt;PAGE #&gt;/&lt;INCLUDE #&gt;:&lt;FILE LINE #&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>VALU</td>
<td>IN PROCESSROWS,</td>
<td>5/259 IN PROCESSROWS, CLASS = LOCAL VAR, TYPE = REAL, OFFSET = 176, LENGTH = 8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TYPE = REAL,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OFFSET = 176,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LENGTH = 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WBNDPTR</td>
<td>IN PROCESSCOLS,</td>
<td>7/338 IN PROCESSCOLS, CLASS = LOCAL VAR, TYPE = POINTER, OFFSET = 224, LENGTH = 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CLASS = LOCAL VAR,</td>
<td>7/362 7/363</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TYPE = POINTER,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OFFSET = 224,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LENGTH = 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WHERE</td>
<td>CLASS = TYPE,</td>
<td>1/26 CLASS = TYPE, TYPE = SCALAR, LENGTH = 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TYPE = SCALAR,</td>
<td>2/86 2/89</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LENGTH = 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WORK</td>
<td>CLASS = LOCAL VAR,</td>
<td>2/76 CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 696, LENGTH = 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TYPE = INTEGER,</td>
<td>5/215 13/659 13/661 13/663</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OFFSET = 696,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LENGTH = 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRB</td>
<td>IN COMPLETE,</td>
<td>10/529 IN COMPLETE, CLASS = PROCEDURE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CLASS = PROCEDURE</td>
<td>12/607 12/608 12/610 12/611 12/613 12/615</td>
<td></td>
</tr>
<tr>
<td>WRNI</td>
<td>IN COMPLETE,</td>
<td>10/523 IN COMPLETE, CLASS = PROCEDURE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CLASS = PROCEDURE</td>
<td>12/609 12/610 12/614 12/615 12/616</td>
<td></td>
</tr>
<tr>
<td>WRITE</td>
<td>PREDEFINED</td>
<td>WRITE PREDEFINED CLASS = PROCEDURE</td>
<td></td>
</tr>
<tr>
<td>WRITELN</td>
<td>PREDEFINED</td>
<td>WRITELN PREDEFINED CLASS = PROCEDURE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6/274 8/386 8/387</td>
<td>9/466 9/475 10/482 10/492 10/499</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12/635 14/701 14/702</td>
<td>14/703 14/704 14/705 14/706 14/707</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14/708 14/711 14/714</td>
<td>14/715 14/717 14/718 14/719 14/720</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14/721 14/722 14/723</td>
<td>14/727 14/728 14/730 14/731 14/732</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14/733 14/734 14/735</td>
<td>14/736</td>
<td></td>
</tr>
<tr>
<td>WRITEOCT</td>
<td>IN DOCONTROL,</td>
<td>WRITEOCT IN DOCONTROL, CLASS = PROCEDURE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CLASS = PROCEDURE</td>
<td>13/644 13/663 13/684 13/688</td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>X IN COEFFOUT, CLASS = LOCAL VAR, TYPE = REAL, OFFSET = 144, LENGTH = 8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3/155 IN COEFFOUT,</td>
<td>4/162 4/166 4/181</td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>X IN WRB, CLASS = LOCAL VAR, TYPE = REAL, OFFSET = 152, LENGTH = 8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10/529 IN WRB,</td>
<td>11/533</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XRBASPTR</td>
<td>XRBASPTR IN PROCESSCOLS, CLASS = LOCAL VAR, TYPE = POINTER, OFFSET = 228, LENGTH = 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7/339 IN PROCESSCOLS,</td>
<td>8/390 9/455 9/459 9/468</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XRBPTR</td>
<td>XRBPTR IN PROCESSROWS, CLASS = LOCAL VAR, TYPE = POINTER, OFFSET = 152, LENGTH = 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5/253 IN PROCESSROWS,</td>
<td>6/309 6/310</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XRHSPTR</td>
<td>XRHSPTR IN PROCESSROWS, CLASS = LOCAL VAR, TYPE = POINTER, OFFSET = 156, LENGTH = 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5/254 IN PROCESSROWS,</td>
<td>6/290 6/291</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XUSER</td>
<td>XUSER CLASS = LOCAL VAR, TYPE = ARRAY, OFFSET = 768, LENGTH = 40</td>
<td></td>
</tr>
</tbody>
</table>
### Cross Reference Listing

<table>
<thead>
<tr>
<th>IDENTIFIER</th>
<th>DEFINITION</th>
<th>ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>3/158 IN COEFFOUT, CLASS = LOCAL VAR, TYPE = REAL, OFFSET = 160, LENGTH = 8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4/162 4/163 4/179</td>
<td></td>
</tr>
<tr>
<td>YBNDPTR</td>
<td>10/519 IN COMPLETE, CLASS = LOCAL VAR, TYPE = POINTER, OFFSET = 156, LENGTH = 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12/626 12/629 12/630</td>
<td></td>
</tr>
<tr>
<td>YRHSPTL</td>
<td>10/517 IN COMPLETE, CLASS = LOCAL VAR, TYPE = POINTER, OFFSET = 148, LENGTH = 4</td>
<td></td>
</tr>
<tr>
<td>ZBNDPTR</td>
<td>10/520 IN COMPLETE, CLASS = LOCAL VAR, TYPE = POINTER, OFFSET = 160, LENGTH = 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12/625 12/627 12/629 12/631</td>
<td></td>
</tr>
<tr>
<td>ZRHSPTL</td>
<td>10/518 IN COMPLETE, CLASS = LOCAL VAR, TYPE = POINTER, OFFSET = 152, LENGTH = 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11/556 11/558</td>
<td></td>
</tr>
</tbody>
</table>

No compiler detected errors.

Options in effect: MARGINS(1,72), SEQ(73,80), XREF(SHORT), LANGLEVEL(EXTENDED), LINCOUNT(60), PAGEWIDTH(128), CHECK, GOSTMT, OPTIMIZE, PXREF, SOURCE, WARNING

Source lines: 751; Compile time: 1.12 seconds; Compile rate: 40232 LPM
<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>ID</th>
<th>ADDR</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAMS2MIP</td>
<td>SD</td>
<td>1</td>
<td>000000</td>
<td>002EA8</td>
</tr>
<tr>
<td>AMPXSTRT</td>
<td>ER</td>
<td>2</td>
<td>000000</td>
<td>000000</td>
</tr>
<tr>
<td>AMPXWTXT</td>
<td>ER</td>
<td>4</td>
<td>000000</td>
<td>000000</td>
</tr>
<tr>
<td>AMPXWCHR</td>
<td>ER</td>
<td>6</td>
<td>000000</td>
<td>000000</td>
</tr>
<tr>
<td>AMPXR</td>
<td>ER</td>
<td>8</td>
<td>000000</td>
<td>000000</td>
</tr>
<tr>
<td>AMPXRLIN</td>
<td>ER</td>
<td>10</td>
<td>000000</td>
<td>000000</td>
</tr>
<tr>
<td>AMPXDISP</td>
<td>ER</td>
<td>12</td>
<td>000000</td>
<td>000000</td>
</tr>
<tr>
<td>AMPXCLS</td>
<td>ER</td>
<td>14</td>
<td>000000</td>
<td>000000</td>
</tr>
<tr>
<td>AMPXBEGIN</td>
<td>LD</td>
<td>0</td>
<td>002CFE</td>
<td>000000</td>
</tr>
<tr>
<td>AMPXOPEN</td>
<td>ER</td>
<td>3</td>
<td>000000</td>
<td>000000</td>
</tr>
<tr>
<td>AMPXLIN</td>
<td>ER</td>
<td>5</td>
<td>000000</td>
<td>000000</td>
</tr>
<tr>
<td>AMPXINT</td>
<td>ER</td>
<td>7</td>
<td>000000</td>
<td>000000</td>
</tr>
<tr>
<td>AMPXRR</td>
<td>ER</td>
<td>9</td>
<td>000000</td>
<td>000000</td>
</tr>
<tr>
<td>AMPXRLIN</td>
<td>ER</td>
<td>11</td>
<td>000000</td>
<td>000000</td>
</tr>
<tr>
<td>AMPXNEW</td>
<td>ER</td>
<td>13</td>
<td>000000</td>
<td>000000</td>
</tr>
</tbody>
</table>

**Source Lines:** 751; **Translate Time:** 0.70 seconds; **Translate Rate:** 64371 LPM

**Total Time:** 2.20 seconds; **Total Rate:** 20482 LPM
NUM := COL - SAVCOL + 1;

IF ((DCTYP = TYP SOS1) AND (NUM < 2)) OR (M < 3))

THEN

PPERROR(4,'TOO FEW VARIABLES IN SOS SET ');

ELSE

SAVBNDREC(COL); (* BUP ETC WRONG, BUT WHO CARES *)

END;

END;

BEGIN

ENVIRON := COLUMN;

SENVIRON := SCOLUMN;

READLN(FDIN);

WRITELN(FMOUT,'COLUMNS');

READLN(FDIN,NNZ,LOWER,LEVEL,UPPER,MARGNFLAG,DCTYP,SOSINDX);

GLOBALSEQ := J;

NEEDBNDREC := (J = 1) OR NOT((LOWER = 0) AND (UPPER = PINF));

IF DOMIP (* CANT CHECK TYPE FLAG HERE *)

THEN

BEGIN

IF DOSOS AND (SOSINDX <> SAVINDX)

THEN (* WRITE RECORD TERMINATING SOS SET *)

END;

BEGIN

IF DOSOS AND (SOSINDX <> SAVINDX)

THEN (* WRITE RECORD TERMINATING SOS SET *)

END;

BEGIN

IF DOSOS AND (SOSINDX <> SAVINDX)

THEN (* WRITE RECORD TERMINATING SOS SET *)

END;

BEGIN

IF DOSOS AND (SOSINDX <> SAVINDX)

THEN (* WRITE RECORD TERMINATING SOS SET *)

END;
THEN
  LOWER := 0.0;
  NEEDBNDREC := TRUE
ELSE (* SOS *)
  BEGIN
  IF NOT((LOWER = 0) AND (UPPER = PINF))
  THEN
    PERROR(9, 'BOUNDS ARE ILLEGAL FOR SOS VARIABLE');
    IF NOT DOSOS
    THEN
      BEGIN
        DOSOS := TRUE;
        SAVINDX := SOSINDX;
        SAVTYP := DCTYP;
        SAVCOL := J;
        NEEDBNDREC := TRUE
      END
  END;
  (* IF MIP *)
  IF NEEDBNDREC
  THEN
    (* SAVE AWAY INFO FOR BOUNDS. LEVEL MAY BE NEEDED FOR NON-LINEAR *)
    (* CODES ALWAYS WRITE AT LEAST ONE BOUND RECORD *)
    SAVBNDREC(J);
    IF NEEDEDBAS
    THEN
      (* DO BASIS *)
      IF MARGNFLAG = 0
      THEN
        IF XRBASEPTR <> NIL
        THEN
          (* WRITE A ROW/COLUMN BASIS RECORD *)
          BEGIN
            WITH XRBASEPTR@ DO BEGIN
              IF UP
              THEN
                WRITE(FBOUT, ' XU ')
              ELSE
                WRITE(FBOUT, ' XL ');
                WRITELN(FBOUT, C':PRC', ':9-PRC', 'R', INDX:1)
            END;
            (* WITH *)
            XRBASEPTR := XRBASEPTR@.INDXNXT;
            NNBROW := NNBROW - 1;
            END
        END; (* IF MIP *)
        IF NEEDBAS
        THEN
          (* WRITE THE NON-ZEROES *)
          ********************************
  ELSE
    (* GAMS SETS THEM EQUAL : NO TOLERANCE CHECKING NEEDED *)
    IF UPPER = LEVEL
    THEN
      WRITELN(FBOUT, ' UL C', J:PRC);
IF (DCTYP = TYPINT) OR (DCTYP = TYPBIN)
THEN
  IF (LOWER <> UPPER)
  THEN
    WRITELN(FMOUT, ':4', 'INT', 'J:PRC', ':7-PRC', 'MARKER', '17', 'INTORG');
  (* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *)
  FOR K := 1 TO NNZ
  DO BEGIN
    READLN(FDIN, ROW, COEFF);
    PRR := PRDIGITS(ROW);
    WRITE(FMOUT, ':4', 'C', 'J:PRC', ':9-PRC', 'R', ROW:PRR, '1:9-PRR);
    COEFFOUT(COEFF, FMOUT);
    WRITELN(FMOUT);
  END;
  (* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *)
IF (DCTYP = TYPINT) OR (DCTYP = TYPBIN)
THEN
  IF (LOWER <> UPPER)
  THEN
    WRITELN(FMOUT, ':4', 'INT', 'A', ':6-PRC', 'MARKER', '17', 'INTEND');
  (* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *)
END;
IF DOSOS
THEN
  DOSOS := TERMSOS(NUMCOLS);
IF NEEDBAS
THEN
  (* GET RID OF THE LIST *)
  DELJBAND(HRBASPTR);
  WRITE(FBOUT, 'ENDATA	');
  CLOSE(FBOUT)
END;

PROCEDURE COMPLETE;
VAR
  PRR : FWID;
  YRHSPTR; ZRHSPTR; TRHSPTR;
  YBDPTR;
  ZBDPTR : TBNDPTR;
  IB : INTEGER;
BEGIN
  PROCEDURE WR(S : TCH2);
BEGIN
  WITH YBDPTR;
  DO WRITELN(FMOUT, ':S', 'BNDNAME', 'C', COLNUM:NDIG)
END;
PROCEDURE WRB(S : TCH2 ; X : REAL);
BEGIN
WITH YBNDPTR@ 00000531 R 10
2 1 I DO WRITE(FMOUT,' ',S,' ',BNDNAME,'C',COLNUM:NDIG,' ':9-NDIG); 00000532 R P 2 10 1 1 1
2 4 I WRITELN(FMOUT) 00000533 3 10 2
1 1 I END; 00000534 P 2
1 1 I BEGIN 00000535 R
1 1 I IF HRHSPTR <> NIL 00000536 R 2 P
1 1 I THEN 00000537 R
1 1 I (* WRITE OUT THE RHS *) 00000538 R 2 P
1 1 I BEGIN 00000539 R
1 1 I ENVIRON := ROW; 00000540 R
1 1 I SENVIRON := SRHS; 00000541 R
1 1 I WRITELN(FMOUT,'RHS'); 00000542 R
1 1 I YRHSPTR := HRHSPTR; 00000543 R
1 1 I WHILE YRHSPTR <> NIL 00000544 R 10 P
1 1 I DO BEGIN 00000545 R
1 1 I WITH YRHSPTR@ 00000546 R 10
1 1 I DO BEGIN 00000547 R
1 1 I GLOBALSEQ := ROWNUM; 00000548 R
1 1 I PRR := PRODIGITS(ROWNUM); 00000549 R
1 1 I WRITE(FMOUT,' ','R',ROWNUM:PRR,' ':'9-PRR); 00000550 R 10 1 1 10 10
1 1 I COEFFOUT(RHSVAL,FMOUT); 00000551 R 3 1 2
1 1 I WRITELN(FMOUT) 00000552 P 2
1 1 I END; 00000553 R
1 1 I ZRHSPTR := YRHSPTR; 00000554 P 10
1 1 I YRHSPTR := YRHSPTR@.ROWNXT; 00000555 R
1 1 I DISPOSE(ZRHSPTR) 00000556 P 10
1 1 I END; 00000557 R
1 1 I HRHSPTR := NIL 00000558 R
1 1 I END 00000559 R
1 1 I ELSE 00000560 P 2
1 1 I (* ALL ZERO RHS: WRITE DUMMY INDIRECT COEFFICIENT FOR APEX4 *) 00000561 R
1 1 I BEGIN 00000562 R
1 1 I WRITELN(FMOUT,'RHS'); 00000563 R
1 1 I WRITELN(FMOUT,' ','R',ROWNUM:PRR,' ':'8,'**IND1') 00000564 R
1 1 I END; 00000565 P 2
1 1 I IF HBNDPTR <> NIL 00000566 P 1
1 1 I THEN 00000567 R
1 1 I (* WRITE OUT THE BOUNDS *) 00000568 R
1 1 I BEGIN 00000569 R
1 1 I ENVIRON := COLUMN; 00000570 R
1 1 I SENVIRON := SBOUND; 00000571 R
1 1 I WRITELN(FMOUT,'BOUNDS'); 00000572 R
1 1 I YBNDPTR := HBNDPTR; 00000573 R
1 1 I WHILE YBNDPTR <> NIL 00000574 R 10 P
1 1 I DO BEGIN 00000575 R
1 1 I WITH YBNDPTR@ 00000576 R 10
1 1 I DO BEGIN 00000577 R
1 1 I GLOBALSEQ := COLNUM; 00000578 R
1 1 I IF BLO = BUP 00000579 R
1 1 I THEN 00000580 R
1 1 I IB := 0 00000581 R 1 1
1 1 I 1B := 0 00000582 R
1 1 I 28 00000583 10
<table>
<thead>
<tr>
<th>SEQ NO</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ELSE</td>
</tr>
<tr>
<td>2</td>
<td>BEGIN</td>
</tr>
<tr>
<td>3</td>
<td>IF BUP = PINF</td>
</tr>
<tr>
<td>4</td>
<td>THEN</td>
</tr>
<tr>
<td>5</td>
<td>IB := 2</td>
</tr>
<tr>
<td>6</td>
<td>ELSE</td>
</tr>
<tr>
<td>7</td>
<td>IF BUP = 0.0</td>
</tr>
<tr>
<td>8</td>
<td>THEN</td>
</tr>
<tr>
<td>9</td>
<td>IB := 0</td>
</tr>
<tr>
<td>10</td>
<td>ELSE</td>
</tr>
<tr>
<td>11</td>
<td>IB := 1;</td>
</tr>
<tr>
<td>12</td>
<td>IF BLO = MINF</td>
</tr>
<tr>
<td>13</td>
<td>THEN</td>
</tr>
<tr>
<td>14</td>
<td>IB := IB + 6</td>
</tr>
<tr>
<td>15</td>
<td>ELSE</td>
</tr>
<tr>
<td>16</td>
<td>IF BLO = 0.0</td>
</tr>
<tr>
<td>17</td>
<td>THEN</td>
</tr>
<tr>
<td>18</td>
<td>IB := IB + 0</td>
</tr>
<tr>
<td>19</td>
<td>ELSE</td>
</tr>
<tr>
<td>20</td>
<td>IB := IB + 3</td>
</tr>
<tr>
<td>21</td>
<td>END;</td>
</tr>
<tr>
<td>22</td>
<td>(* CONTINUOUS AND INTEGER VARIABLES *)</td>
</tr>
<tr>
<td>23</td>
<td>CASE IB OF</td>
</tr>
<tr>
<td>24</td>
<td>0: (* FX *) WRB('FX',BLO)</td>
</tr>
<tr>
<td>25</td>
<td>1: (* 0 NZ *) WRB('UP',BUP)</td>
</tr>
<tr>
<td>26</td>
<td>2: (* 0 INF *) WRB('PL')</td>
</tr>
<tr>
<td>27</td>
<td>3: (* APEX 4 *) BEGIN WRB('MI'); WRB('LO',BLO) END;</td>
</tr>
<tr>
<td>28</td>
<td>4: (* NZ NZ *) BEGIN WRB('LO',BLO); WRB('UP',BUP)</td>
</tr>
<tr>
<td>29</td>
<td>5: (* NZ INF *) WRB('LO',BLO)</td>
</tr>
<tr>
<td>30</td>
<td>6: (* -INF 0 *) WRB('MI')</td>
</tr>
<tr>
<td>31</td>
<td>7: (* -INF NZ *) BEGIN WRB('MI'); WRB('UP',BUP) END;</td>
</tr>
<tr>
<td>32</td>
<td>8: (* -INF INF *) WRB('FR')</td>
</tr>
<tr>
<td>33</td>
<td>END;</td>
</tr>
<tr>
<td>34</td>
<td>END;</td>
</tr>
<tr>
<td>35</td>
<td>(* WITH *)</td>
</tr>
<tr>
<td>36</td>
<td>IF DOMIP</td>
</tr>
<tr>
<td>37</td>
<td>THEN</td>
</tr>
<tr>
<td>38</td>
<td>IF (<a href="mailto:YBNDPTR@.MIPTYP">YBNDPTR@.MIPTYP</a> = TYPSOS1) OR (<a href="mailto:YBNDPTR@.MIPTYP">YBNDPTR@.MIPTYP</a> = TYPSOS2)</td>
</tr>
<tr>
<td>39</td>
<td>THEN</td>
</tr>
<tr>
<td>40</td>
<td>BEGIN</td>
</tr>
<tr>
<td>41</td>
<td>ZBNDPTR := YBNDPTR;</td>
</tr>
<tr>
<td>42</td>
<td>YBNDPTR := <a href="mailto:YBNDPTR@.BNDNXT">YBNDPTR@.BNDNXT</a>;</td>
</tr>
<tr>
<td>43</td>
<td>DISPOSE(ZBNDPTR)</td>
</tr>
<tr>
<td>44</td>
<td>END;</td>
</tr>
<tr>
<td>45</td>
<td>(* WHILE *)</td>
</tr>
<tr>
<td>46</td>
<td>HBDNTR := NIL</td>
</tr>
<tr>
<td>47</td>
<td>END;</td>
</tr>
<tr>
<td>48</td>
<td>(* IF *)</td>
</tr>
<tr>
<td>49</td>
<td>WRITELN(FMOUT,'ENDATA');</td>
</tr>
<tr>
<td>50</td>
<td>CLOSE(FMOUT);</td>
</tr>
</tbody>
</table>
CLOSE(FDIN)
END;

PROCEDURE DOCONTROL;
(* WRITE CONTROL INFORMATION. IF ERRORS CONTROL INFO SHOULD BY-PASS *)
(* SOLVE AND POST-PROCESSING AND RETURN CONTROL TO GAMS *)

procedure writeoct(var f : text; e,e2 : integer);
begin
end;

PROCEDURE PUTRFL;
(* CYBER ONLY - WRITE CORE REQUEST FOR APEX4 *)

CONST
MAXFL = 125000; (* LEAVE A LITTLE FOR CMM *)

VAR
FL,
ROC,
RECFI,
MINFL : REAL;

BEGIN
IF WORK > 0 THEN
IF WORK < MAXFL THEN
WRITEOCT(FCOUT,WORK,6)
ELSE
BEGIN
ENVIRON := GLOBAL;
PPERROR(0,'VALUE OF OPTION WORK TOO LARGE');
END
ELSE
BEGIN
ROC := NUMROWS/NUMCOLS;
IF ROC > 1.0 THEN
ROC := 1.0/ROC;
MINFL := 3.45*NUMROWS + 0.5*NUMCOLS + 0.15*NUMNNZ
+ 0.5*NUMNNZ*ROC + 33000;
RECFI := MINFL + 3.75*NUMNNZ*ROC;
IF RECFI < MAXFL THEN
IF RECFI > 36864 THEN
WRITEOCT(FCOUT,ROUND(RECFI),6)
ELSE
WRITEOCT(FCOUT,RECFI,6)
ELSE
WRITEOCT(FCOUT,MAXFL,6)
ELSE
WRITEOCT(FCOUT,MAXFL,6)
END
IF MINFL < MAXFL THEN
WRITEOCT(FCOUT,MINFL,6)
END
```pascal
BEGIN
ENVIRO := GLOBAL;
BEGIN
PPERROR(0, 'NOT ENOUGH CM TO SOLVE MODEL');
END;
(* PUTRFL *)
REWRITE(FCOUT);
WRITELN(FCOUT, ':9, 'PROGRAM');
WRITELN(FCOUT, ':9, 'INITIALZ');
WRITELN(FCOUT, ':9, 'MOVE(XDATA, ',GAMS''));
WRITELN(FCOUT, ':9, 'MOVE(XPBNAME, ',ANDELU''));
WRITELN(FCOUT, ':9, 'MOVE(XCORE=1000000)');
WRITELN(FCOUT, ':9, 'CONVERT(''SUMM''4, ',SCRA''));
WRITELN(FCOUT, ':9, 'MOVE(XBOUND, ',GBND''));
BEGIN
IF (IDIR=0) THEN
BEGIN
WRITELN(FCOUT, ':9, 'SETUP(''MIN''));
ELSE
WRITELN(FCOUT, ':9, 'SETUP(''MAX''));
END;
WRITELN(FCOUT, ':9, 'OPTIMIX); END;
BEGIN
WRITELN(FCOUT, ':9, 'XITERNO=0');
WRITELN(FCOUT, ':9, 'OPTIMIX'); END;
BEGIN
WRITELN(FCOUT, ':9, 'SOLUTION(''FILE'', ',DPLUSER1''));
WRITELN(FCOUT, ':9, 'EXIT');
END;
BEGIN
CLOSE(FCOUT); END;
END;
END;
END;
END;
```

BEGIN PROCESSROWS;

PROCESSCOLS;

COMPLETE

END;

IDOCONTROL
END.
<table>
<thead>
<tr>
<th>IDENTIFIER</th>
<th>DEFINITION</th>
<th>ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>PREDEFINED</td>
<td>CLASS = FUNCTION</td>
</tr>
<tr>
<td>BLO</td>
<td>1/40</td>
<td>IN TBND, CLASS = FIELD, TYPE = REAL, OFFSET = 0, LENGTH = 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/351, 11/581, 12/595, 12/599, 12/607, 12/610, 12/611, 12/613</td>
</tr>
<tr>
<td>BLVL</td>
<td>1/41</td>
<td>IN TBND, CLASS = FIELD, TYPE = REAL, OFFSET = 8, LENGTH = 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/352</td>
</tr>
<tr>
<td>BNDNAME</td>
<td>1/14</td>
<td>CLASS = CONSTANT, TYPE = ARRAY</td>
</tr>
<tr>
<td>BNDNXT</td>
<td>1/46</td>
<td>IN TBND, CLASS = FIELD, TYPE = POINTER, OFFSET = 36, LENGTH = 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/356, 7/362, 12/626, 12/630</td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>PREDEFINED</td>
<td>CLASS = TYPE, TYPE = BOOLEAN, LENGTH = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1/53, 2/105, 7/341, 7/366</td>
</tr>
<tr>
<td>BPTR</td>
<td>5/233</td>
<td>IN ELIMBLIST, CLASS = VAR PARAMETER, TYPE = POINTER, OFFSET = 144, LENGTH = 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/239, 5/246</td>
</tr>
<tr>
<td>BRATIO</td>
<td>2/95</td>
<td>CLASS = LOCAL VAR, TYPE = REAL, OFFSET = 832, LENGTH = 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/217, 6/314</td>
</tr>
<tr>
<td>BUP</td>
<td>1/42</td>
<td>IN TBND, CLASS = FIELD, TYPE = REAL, OFFSET = 16, LENGTH = 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/353, 11/581, 12/586, 12/590, 12/608, 12/611, 12/615</td>
</tr>
<tr>
<td>B1PTR</td>
<td>5/236</td>
<td>IN ELIMBLIST, CLASS = LOCAL VAR, TYPE = POINTER, OFFSET = 148, LENGTH = 4</td>
</tr>
<tr>
<td>B2PTR</td>
<td>5/237</td>
<td>IN ELIMBLIST, CLASS = LOCAL VAR, TYPE = POINTER, OFFSET = 152, LENGTH = 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/242, 5/244</td>
</tr>
<tr>
<td>C</td>
<td>4/188</td>
<td>IN INITIALIZE, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 148, LENGTH = 4</td>
</tr>
<tr>
<td>CHAR</td>
<td>PREDEFINED</td>
<td>CLASS = TYPE, TYPE = CHAR, LENGTH = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1/25, 2/57, 2/89, 2/107, 5/251</td>
</tr>
<tr>
<td>CLOSE</td>
<td>3/109</td>
<td>CLASS = PROCEDURE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/222, 10/511, 12/636, 12/637, 14/737</td>
</tr>
<tr>
<td>COEFF</td>
<td>7/322</td>
<td>IN PROCESSCOLS, CLASS = LOCAL VAR, TYPE = REAL, OFFSET = 144, LENGTH = 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10/407, 10/491</td>
</tr>
<tr>
<td>COEFFOUT</td>
<td>3/155</td>
<td>CLASS = PROCEDURE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10/491, 11/533, 11/553</td>
</tr>
<tr>
<td>COL</td>
<td>7/343</td>
<td>IN SAVBNDREC, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 144, LENGTH = 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/354</td>
</tr>
<tr>
<td>IDENTIFIER</td>
<td>DEFINITION</td>
<td>ATTRIBUTES</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>COL</td>
<td>7/366 IN TERMSOS, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 148, LENGTH = 4</td>
<td>7/372 8/378</td>
</tr>
<tr>
<td>COLNUM</td>
<td>1/44 IN TBND, CLASS = FIELD, TYPE = INTEGER, OFFSET = 28, LENGTH = 4</td>
<td>7/354 10/526 11/532 11/580</td>
</tr>
<tr>
<td>COLUMN</td>
<td>1/26 CLASS = CONSTANT, TYPE = SCALAR, VALUE = 1</td>
<td>4/199 8/383 11/572</td>
</tr>
<tr>
<td>COMP</td>
<td>4/187 IN INITIALIZE, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 144, LENGTH = 4</td>
<td>4/205 4/206</td>
</tr>
<tr>
<td>COMPLETE</td>
<td>10/514 CLASS = PROCEDURE</td>
<td>15/748</td>
</tr>
<tr>
<td>DCTYP</td>
<td>7/329 IN PROCESSCOLS, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 188, LENGTH = 4</td>
<td>7/355 8/373 8/393 8/404 8/406 8/418 9/439</td>
</tr>
<tr>
<td>DISPOSE</td>
<td>PREDEFINED CLASS = PROCEDURE</td>
<td>5/244 11/558 12/627 12/631</td>
</tr>
<tr>
<td>DOCONTROL</td>
<td>13/640 CLASS = PROCEDURE</td>
<td>15/750</td>
</tr>
<tr>
<td>DOMIP</td>
<td>2/102 CLASS = LOCAL VAR, TYPE = BOOLEAN, OFFSET = 857, LENGTH = 1</td>
<td>4/206 8/397 12/619 14/724</td>
</tr>
<tr>
<td>DOMLIM</td>
<td>2/79 CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 708, LENGTH = 4</td>
<td>5/216</td>
</tr>
<tr>
<td>DTXT</td>
<td>2/107 CLASS = LOCAL VAR, TYPE = ARRAY, OFFSET = 864, LENGTH = 6</td>
<td>4/200</td>
</tr>
<tr>
<td>EFORMATOK</td>
<td>1/12 CLASS = CONSTANT, TYPE = BOOLEAN, VALUE = 0</td>
<td>4/167</td>
</tr>
<tr>
<td>ELIMBLIST</td>
<td>5/233 CLASS = PROCEDURE</td>
<td>6/317 10/509</td>
</tr>
<tr>
<td>ERRMKR</td>
<td>2/89 CLASS = LOCAL VAR, TYPE = ARRAY, OFFSET = 740, LENGTH = 3</td>
<td>3/130 3/134 4/199</td>
</tr>
<tr>
<td>ERRNUM</td>
<td>3/113 IN PPERROR, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 144, LENGTH = 4</td>
<td>3/135</td>
</tr>
<tr>
<td>IDENTIFIER</td>
<td>DEFINITION</td>
<td>ATTRIBUTES</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>ERRORFREE</td>
<td>2/103</td>
<td>CLASS = LOCAL VAR, TYPE = BOOLEAN, OFFSET = 858, LENGTH = 1</td>
</tr>
<tr>
<td>ERRTXT</td>
<td>1/25</td>
<td>CLASS = TYPE, TYPE = ARRAY, LENGTH = 50</td>
</tr>
<tr>
<td>F</td>
<td>3/155</td>
<td>IN COEFFOUT, CLASS = VAR PARAMETER, TYPE = FILE, OFFSET = 152, LENGTH = 64</td>
</tr>
<tr>
<td>FALSE</td>
<td>PREDEFINED</td>
<td>CLASS = CONSTANT, TYPE = BOOLEAN, VALUE = 0</td>
</tr>
<tr>
<td>FBOUT</td>
<td>2/72</td>
<td>CLASS = LOCAL VAR, TYPE = FILE, OFFSET = 540, LENGTH = 64</td>
</tr>
<tr>
<td>FCIN</td>
<td>2/68</td>
<td>CLASS = LOCAL VAR, TYPE = FILE, OFFSET = 284, LENGTH = 64</td>
</tr>
<tr>
<td>FCOUT</td>
<td>2/66</td>
<td>CLASS = LOCAL VAR, TYPE = FILE, OFFSET = 156, LENGTH = 64</td>
</tr>
<tr>
<td>FDIN</td>
<td>2/67</td>
<td>CLASS = LOCAL VAR, TYPE = FILE, OFFSET = 220, LENGTH = 64</td>
</tr>
<tr>
<td>FMOUT</td>
<td>2/71</td>
<td>CLASS = LOCAL VAR, TYPE = FILE, OFFSET = 476, LENGTH = 64</td>
</tr>
<tr>
<td>FSTAOUT</td>
<td>2/69</td>
<td>CLASS = LOCAL VAR, TYPE = FILE, OFFSET = 348, LENGTH = 64</td>
</tr>
<tr>
<td>FWID</td>
<td>1/23</td>
<td>CLASS = TYPE, TYPE = INTEGER, LENGTH = 4</td>
</tr>
<tr>
<td>G</td>
<td>3/141</td>
<td>IN PRDIGITS, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 152, LENGTH = 4</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>1/26</td>
<td>CLASS = CONSTANT, TYPE = SCALAR, VALUE = 0</td>
</tr>
<tr>
<td>GLOBALSEQ</td>
<td>2/88</td>
<td>CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 736, LENGTH = 4</td>
</tr>
<tr>
<td>HBDPTR</td>
<td>2/63</td>
<td>CLASS = LOCAL VAR, TYPE = POINTER, OFFSET = 148, LENGTH = 4</td>
</tr>
<tr>
<td>IDENTIFIER</td>
<td>DEFINITION</td>
<td>ATTRIBUTES</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>HRBASPTR</td>
<td>2/65</td>
<td>CLASS = LOCAL VAR, TYPE = POINTER, OFFSET = 152, LENGTH = 4</td>
</tr>
<tr>
<td>HRHS PTR</td>
<td>2/61</td>
<td>CLASS = LOCAL VAR, TYPE = POINTER, OFFSET = 144, LENGTH = 4</td>
</tr>
<tr>
<td>I</td>
<td>3/160</td>
<td>IN COEFFOUT, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 172, LENGTH = 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4/178</td>
</tr>
<tr>
<td>IB</td>
<td>10/521</td>
<td>IN COMPLETE, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 164, LENGTH = 4</td>
</tr>
<tr>
<td>IDIR</td>
<td>2/106</td>
<td>CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 860, LENGTH = 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4/212</td>
</tr>
<tr>
<td>INDX</td>
<td>1/52</td>
<td>IN TBAS, CLASS = FIELD, TYPE = INTEGER, OFFSET = 0, LENGTH = 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6/300</td>
</tr>
<tr>
<td>INDXNXT</td>
<td>1/54</td>
<td>IN TBAS, CLASS = FIELD, TYPE = POINTER, OFFSET = 8, LENGTH = 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/243</td>
</tr>
<tr>
<td>INITIALIZE</td>
<td>4/185</td>
<td>CLASS = PROCEDURE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14/742</td>
</tr>
<tr>
<td>INTEGER</td>
<td>PREDEFINED</td>
<td>CLASS = TYPE, TYPE = INTEGER, LENGTH = 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13/644</td>
</tr>
<tr>
<td>IPOWEROF10</td>
<td>2/75</td>
<td>CLASS = LOCAL VAR, TYPE = ARRAY, OFFSET = 680, LENGTH = 16</td>
</tr>
<tr>
<td>ITERLIM</td>
<td>2/78</td>
<td>CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 704, LENGTH = 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/214</td>
</tr>
<tr>
<td>ITYP</td>
<td>5/258</td>
<td>IN PROCESSROWS, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 172, LENGTH = 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6/273</td>
</tr>
<tr>
<td>IUSER</td>
<td>2/90</td>
<td>CLASS = LOCAL VAR, TYPE = ARRAY, OFFSET = 744, LENGTH = 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/219</td>
</tr>
<tr>
<td>J</td>
<td>5/256</td>
<td>IN PROCESSROWS, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 164, LENGTH = 4</td>
</tr>
<tr>
<td>J</td>
<td>7/337</td>
<td>IN PROCESSCOLS, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 220, LENGTH = 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9/475</td>
</tr>
<tr>
<td>K</td>
<td>7/335</td>
<td>IN PROCESSCOLS, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 212, LENGTH = 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10/485</td>
</tr>
<tr>
<td>K</td>
<td>3/138</td>
<td>IN PRODIGITS, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 148, LENGTH = 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/143</td>
</tr>
<tr>
<td>IDENTIFIER</td>
<td>DEFINITION</td>
<td>ATtributes</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LEVEL</td>
<td>7/324</td>
<td>IN PROCESSCOLS, CLASS = LOCAL VAR, TYPE = REAL, OFFSET = 160, LENGTH = 8</td>
</tr>
<tr>
<td></td>
<td>7/352</td>
<td></td>
</tr>
<tr>
<td>LOWER</td>
<td>7/323</td>
<td>IN PROCESSCOLS, CLASS = LOCAL VAR, TYPE = REAL, OFFSET = 152, LENGTH = 8</td>
</tr>
<tr>
<td></td>
<td>7/351</td>
<td></td>
</tr>
<tr>
<td>MARGINFLAG</td>
<td>7/332</td>
<td>IN PROCESSCOLS, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 200, LENGTH = 4</td>
</tr>
<tr>
<td></td>
<td>8/393</td>
<td></td>
</tr>
<tr>
<td>MARGINFLAG</td>
<td>5/257</td>
<td>IN PROCESSROWS, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 168, LENGTH = 4</td>
</tr>
<tr>
<td></td>
<td>6/273</td>
<td></td>
</tr>
<tr>
<td>MAXFL</td>
<td>13/651</td>
<td>IN PUTRFL, CLASS = CONSTANT, TYPE = INTEGER, VALUE = 125000</td>
</tr>
<tr>
<td></td>
<td>13/661</td>
<td></td>
</tr>
<tr>
<td>MAXINT</td>
<td>PREDEFINED</td>
<td>CLASS = CONSTANT, TYPE = INTEGER, VALUE = 2147483647</td>
</tr>
<tr>
<td></td>
<td>1/21</td>
<td></td>
</tr>
<tr>
<td>MAXRC</td>
<td>1/16</td>
<td>CLASS = CONSTANT, TYPE = INTEGER, VALUE = 32767</td>
</tr>
<tr>
<td></td>
<td>1/22</td>
<td></td>
</tr>
<tr>
<td>MINFO</td>
<td>2/93</td>
<td>CLASS = LOCAL VAR, TYPE = REAL, OFFSET = 816, LENGTH = 8</td>
</tr>
<tr>
<td></td>
<td>5/224</td>
<td></td>
</tr>
<tr>
<td>MINFL</td>
<td>13/656</td>
<td>IN PUTRFL, CLASS = LOCAL VAR, TYPE = REAL, OFFSET = 168, LENGTH = 8</td>
</tr>
<tr>
<td></td>
<td>13/675</td>
<td></td>
</tr>
<tr>
<td>MIPTYP</td>
<td>1/45</td>
<td>IN TBND, CLASS = FIELD, TYPE = INTEGER, OFFSET = 32, LENGTH = 4</td>
</tr>
<tr>
<td></td>
<td>7/355</td>
<td></td>
</tr>
<tr>
<td>NBBNDPTR</td>
<td>7/345</td>
<td>IN SAVBNDREC, CLASS = LOCAL VAR, TYPE = POINTER, OFFSET = 148, LENGTH = 4</td>
</tr>
<tr>
<td></td>
<td>7/347</td>
<td></td>
</tr>
<tr>
<td>NDIG</td>
<td>1/43</td>
<td>IN TBND, CLASS = FIELD, TYPE = INTEGER, OFFSET = 24, LENGTH = 4</td>
</tr>
<tr>
<td></td>
<td>7/350</td>
<td></td>
</tr>
<tr>
<td>NEEDBAS</td>
<td>2/105</td>
<td>CLASS = LOCAL VAR, TYPE = BOOLEAN, OFFSET = 859, LENGTH = 1</td>
</tr>
<tr>
<td></td>
<td>6/314</td>
<td></td>
</tr>
<tr>
<td>NEEDBNDREC</td>
<td>7/341</td>
<td>IN PROCESSCOLS, CLASS = LOCAL VAR, TYPE = BOOLEAN, OFFSET = 233, LENGTH = 1</td>
</tr>
<tr>
<td></td>
<td>8/396</td>
<td></td>
</tr>
<tr>
<td>NEW</td>
<td>PREDEFINED</td>
<td>CLASS = PROCEDURE</td>
</tr>
<tr>
<td></td>
<td>6/279</td>
<td></td>
</tr>
<tr>
<td>NIL</td>
<td>PREDEFINED</td>
<td>CLASS = CONSTANT, TYPE = POINTER</td>
</tr>
<tr>
<td></td>
<td>5/240</td>
<td></td>
</tr>
<tr>
<td>NNBROW</td>
<td>2/81</td>
<td>CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 716, LENGTH = 4</td>
</tr>
<tr>
<td>IDENTIFIER</td>
<td>DEFINITION</td>
<td>ATTRIBUTES</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>NNZ</td>
<td>IN PROCESSCOLS, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 204, LENGTH = 4</td>
<td>5/264 6/311 6/314 9/469</td>
</tr>
<tr>
<td>NOTUSED</td>
<td>CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 712, LENGTH = 4</td>
<td>2/80 4/205 4/210</td>
</tr>
<tr>
<td>NUM</td>
<td>IN TERMSOS, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 152, LENGTH = 4</td>
<td>7/370 7/372 8/373 8/374</td>
</tr>
<tr>
<td>NUMCOLS</td>
<td>CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 724, LENGTH = 4</td>
<td>2/83 4/208 5/228 8/391 10/505 13/671 13/675</td>
</tr>
<tr>
<td>NUMNNZ</td>
<td>CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 728, LENGTH = 4</td>
<td>2/84 4/209 13/675 13/676 13/677</td>
</tr>
<tr>
<td>OBJROSEQ</td>
<td>CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 700, LENGTH = 4</td>
<td>2/77 4/211 14/716</td>
</tr>
<tr>
<td>PINF</td>
<td>CLASS = LOCAL VAR, TYPE = REAL, OFFSET = 808, LENGTH = 8</td>
<td>2/92 5/224 8/396 8/421 9/431 12/586</td>
</tr>
<tr>
<td>POSINT</td>
<td>CLASS = TYPE, TYPE = INTEGER, LENGTH = 4</td>
<td>1/21 3/138 3/141 7/335</td>
</tr>
<tr>
<td>PROCESSCOLS</td>
<td>CLASS = PROCEDURE</td>
<td>7/320 8/395 10/488 11/551 14/716</td>
</tr>
<tr>
<td>PROCESSROWS</td>
<td>CLASS = PROCEDURE</td>
<td>5/249 15/747</td>
</tr>
<tr>
<td>PRR</td>
<td>IN PROCESSCOLS, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 176, LENGTH = 4</td>
<td>7/326 15/746</td>
</tr>
</tbody>
</table>
**CROSS REFERENCE LISTING**

<table>
<thead>
<tr>
<th>IDENTIFIER</th>
<th>DEFINITION</th>
<th>ATTRIBUTES</th>
<th>FORMAT: &lt;PAGE #&gt;/&lt;INCLUDE #&gt;:&lt;FILE LINE #&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRR</td>
<td>IN COMPLETE, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 144, LENGTH = 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCSEQT</td>
<td>CLASS = TYPE, TYPE = INTEGER, LENGTH = 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>READ</td>
<td>PREDEFINED, CLASS = PROCEDURE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>READLN</td>
<td>PREDEFINED, CLASS = PROCEDURE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REAL</td>
<td>PREDEFINED, CLASS = TYPE, TYPE = REAL, LENGTH = 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECFL</td>
<td>IN PUTRFL, CLASS = LOCAL VAR, TYPE = REAL, OFFSET = 160, LENGTH = 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESET</td>
<td>PREDEFINED, CLASS = PROCEDURE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESLIM</td>
<td>CLASS = LOCAL VAR, TYPE = REAL, OFFSET = 824, LENGTH = 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REWRITE</td>
<td>PREDEFINED, CLASS = PROCEDURE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RHSNAME</td>
<td>CLASS = CONSTANT, TYPE = ARRAY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RHSVAL</td>
<td>IN TRHS, CLASS = FIELD, TYPE = REAL, OFFSET = 0, LENGTH = 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROC</td>
<td>IN PUTRFL, CLASS = LOCAL VAR, TYPE = REAL, OFFSET = 152, LENGTH = 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROUND</td>
<td>PREDEFINED, CLASS = FUNCTION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROW</td>
<td>IN PROCESSCOLS, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 208, LENGTH = 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROW</td>
<td>CLASS = CONSTANT, TYPE = SCALAR, VALUE = 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROWNUM</td>
<td>IN TRHS, CLASS = FIELD, TYPE = INTEGER, OFFSET = 8, LENGTH = 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IDENTIFIER</td>
<td>DEFINITION</td>
<td>ATTRIBUTES</td>
<td>FORMAT: &lt;PAGE #&gt;/&lt;INCLUDE #&gt;:&lt;FILE LINE #&gt;</td>
</tr>
<tr>
<td>----------------</td>
<td>------------</td>
<td>------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>ROWNXT</td>
<td>1/34</td>
<td>IN TRHS, CLASS = FIELD, TYPE = POINTER, OFFSET = 12, LENGTH = 4</td>
<td>6/282/11/550/11/551/11/552</td>
</tr>
<tr>
<td>RPOWEROF10</td>
<td>2/74</td>
<td>CLASS = LOCAL VAR, TYPE = ARRAY, OFFSET = 608, LENGTH = 72</td>
<td>4/163/4/179/4/190/4/192</td>
</tr>
<tr>
<td>RST</td>
<td>5/251</td>
<td>IN PROCESSROWS, CLASS = LOCAL VAR, TYPE = ARRAY, OFFSET = 144, LENGTH = 4</td>
<td>5/261/6/274</td>
</tr>
<tr>
<td>S</td>
<td>10/523</td>
<td>IN WRI, CLASS = LOCAL VAR, TYPE = ARRAY, OFFSET = 148, LENGTH = 2</td>
<td>10/526</td>
</tr>
<tr>
<td>S</td>
<td>10/529</td>
<td>IN WRB, CLASS = LOCAL VAR, TYPE = ARRAY, OFFSET = 160, LENGTH = 2</td>
<td>11/532</td>
</tr>
<tr>
<td>SAVBDNREC</td>
<td>7/343</td>
<td>IN PROCESSCOLS, CLASS = PROCEDURE</td>
<td>8/378/9/449</td>
</tr>
<tr>
<td>SAVCOL</td>
<td>7/336</td>
<td>IN PROCESSCOLS, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 216, LENGTH = 4</td>
<td>7/372/9/440</td>
</tr>
<tr>
<td>SAVINDX</td>
<td>7/330</td>
<td>IN PROCESSCOLS, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 192, LENGTH = 4</td>
<td>8/400/9/438</td>
</tr>
<tr>
<td>SAVTYP</td>
<td>7/328</td>
<td>IN PROCESSCOLS, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 184, LENGTH = 4</td>
<td>9/439</td>
</tr>
<tr>
<td>SBOUND</td>
<td>1/27</td>
<td>CLASS = CONSTANT, TYPE = SCALAR, VALUE = 2</td>
<td>4/173/11/573</td>
</tr>
<tr>
<td>SCOLUMN</td>
<td>1/27</td>
<td>CLASS = CONSTANT, TYPE = SCALAR, VALUE = 1</td>
<td>4/171/8/384</td>
</tr>
<tr>
<td>SOSINDX</td>
<td>7/331</td>
<td>IN PROCESSCOLS, CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 196, LENGTH = 4</td>
<td>8/393/8/400/9/438</td>
</tr>
<tr>
<td>SRHS</td>
<td>1/27</td>
<td>CLASS = CONSTANT, TYPE = SCALAR, VALUE = 3</td>
<td>4/172/11/543</td>
</tr>
<tr>
<td>SROW</td>
<td>1/27</td>
<td>CLASS = CONSTANT, TYPE = SCALAR, VALUE = 0</td>
<td>4/170/6/269</td>
</tr>
<tr>
<td>SUBWHERE</td>
<td>1/27</td>
<td>CLASS = TYPE, TYPE = SCALAR, LENGTH = 1</td>
<td>2/87</td>
</tr>
<tr>
<td>SYSOUT</td>
<td>2/101</td>
<td>CLASS = LOCAL VAR, TYPE = BOOLEAN, OFFSET = 856, LENGTH = 1</td>
<td>5/215</td>
</tr>
<tr>
<td>IDENTIFIER</td>
<td>DEFINITION</td>
<td>ATTRIBUTES</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td></td>
</tr>
<tr>
<td>TBAS</td>
<td>CLASS = TYPE, TYPE = RECORD, LENGTH = 12</td>
<td>1/50</td>
<td></td>
</tr>
<tr>
<td>TBND</td>
<td>CLASS = TYPE, TYPE = RECORD, LENGTH = 40</td>
<td>1/38</td>
<td></td>
</tr>
<tr>
<td>TBNDPTR</td>
<td>CLASS = TYPE, TYPE = POINTER, LENGTH = 4</td>
<td>1/46 2/63 7/338 7/345 10/520</td>
<td></td>
</tr>
<tr>
<td>TCH2</td>
<td>CLASS = TYPE, TYPE = ARRAY, LENGTH = 2</td>
<td>10/523 10/529</td>
<td></td>
</tr>
<tr>
<td>TERMSOS</td>
<td>IN PROCESSCOLS, CLASS = FUNCTION, TYPE = BOOLEAN, LENGTH = 1</td>
<td>8/379 8/1403 10/505</td>
<td></td>
</tr>
<tr>
<td>TEXT</td>
<td>PREDEFINED CLASS = TYPE, TYPE = FILE, LENGTH = 64</td>
<td>2/73 3/109 3/155 13/644</td>
<td></td>
</tr>
<tr>
<td>TMIP</td>
<td>CLASS = TYPE, TYPE = INTEGER, LENGTH = 4</td>
<td>1/45 2/100 7/329</td>
<td></td>
</tr>
<tr>
<td>TRHS</td>
<td>CLASS = TYPE, TYPE = RECORD, LENGTH = 16</td>
<td>1/30</td>
<td></td>
</tr>
<tr>
<td>TRHSPTR</td>
<td>CLASS = TYPE, TYPE = POINTER, LENGTH = 4</td>
<td>1/34 2/61 5/255 10/518</td>
<td></td>
</tr>
<tr>
<td>TRUE</td>
<td>PREDEFINED CLASS = CONSTANT, TYPE = BOOLEAN, VALUE = 1</td>
<td>4/197 8/415 9/427 9/437 9/441</td>
<td></td>
</tr>
<tr>
<td>TXT</td>
<td>IN PPERROR, CLASS = LOCAL VAR, TYPE = ARRAY, OFFSET = 152, LENGTH = 50</td>
<td>3/113 3/131 3/135</td>
<td></td>
</tr>
<tr>
<td>TYPBIN</td>
<td>CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 840, LENGTH = 4</td>
<td>4/205 8/406 9/478 10/495</td>
<td></td>
</tr>
<tr>
<td>TYPINT</td>
<td>CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 844, LENGTH = 4</td>
<td>4/205 8/418 9/478 10/495</td>
<td></td>
</tr>
<tr>
<td>TYP SOS1</td>
<td>CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 848, LENGTH = 4</td>
<td>4/205 8/373 12/621</td>
<td></td>
</tr>
<tr>
<td>TYP SOS2</td>
<td>CLASS = LOCAL VAR, TYPE = INTEGER, OFFSET = 852, LENGTH = 4</td>
<td>4/205 8/374 12/622</td>
<td></td>
</tr>
<tr>
<td>UP</td>
<td>IN TBAS, CLASS = FIELD, TYPE = BOOLEAN, OFFSET = 4, LENGTH = 1</td>
<td>6/302 9/461</td>
<td></td>
</tr>
<tr>
<td>UPPER</td>
<td>IN PROCESSCOLS, CLASS = LOCAL VAR, TYPE = REAL, OFFSET = 168, LENGTH = 8</td>
<td>7/325</td>
<td></td>
</tr>
</tbody>
</table>
May 10, 1986

Dr. Richard S. Barr
School of Engineering and Applied Science
Operations Research Department
Southern Methodist University
Dallas, Texas 75275

Dear Dr. Barr:

I submit the accompanying report entitled "GAMS/MPSX INTERFACE" as part of the requirements for OREM 4390, Senior Design.

The report presents an introduction to GAMS, a discussion on integer programming, and a description of what has already been done, and what needs to be done to complete an interface between GAMS and MPSX/370, specifically the Mixed Integer Programming/370 package.

I would specifically like to thank you, Gary Slacdek of the computer center User's Services, and Steve, the graduate student down at UT-Austin, who all gave me a great deal of help on this project.

Sincerely,

Lawrence E. McPhail
OREM 4390